1. Data Loading and Exploration

- **Head:** `import numpy as np`
- **Tail:** `data.shape`

Explanation:
This part imports necessary libraries like NumPy, Pandas, Matplotlib, and scikit-learn. The dataset `Social_Network_Ads.csv` is loaded into a DataFrame. Initial exploration functions like `.head()`, `.info()`, `.describe()`, `.isnull().sum()`, and `.shape` are used to understand the structure, data types, missing values, and shape of the dataset.

---

2. Feature Selection

- **Head:** `x = data.iloc[:,2:4]`
- **Tail:** `y = data.iloc[:,4]`

Explanation:
Here, feature selection is performed by extracting the `Age` and `EstimatedSalary` columns as features (`x`) and `Purchased` as the target variable (`y`). This step reduces the dataset to relevant attributes for model training.

---

3. Train-Test Split

- **Head:** `x_train, x_test, y_train, y_test = train_test_split(...)`
- **Tail:** `random_state=42`

Explanation:
The dataset is split into training and testing subsets with a 75%-25% ratio using `train_test_split`. A random seed is set for reproducibility of results.

---

4. Feature Scaling

- **Head:** `scale = StandardScaler()`
- **Tail:** `x_test = scale.transform(x_test)`

Explanation:
Standardization is applied using `StandardScaler` to scale features to have a mean of 0 and standard deviation of 1. This helps the logistic regression model converge faster and perform better.

---

5. Model Training

- **Head:** `lr = LogisticRegression(...)`
- **Tail:** `lr.fit(x_train,y_train)`

Explanation:
A Logistic Regression model is created with the `lbfgs` solver and random state. The model is trained (fit) on the scaled training data.

---

6. Prediction

- **Head:** `pred = lr.predict(x_test)`
- **Tail:** `print('Predicted Output:\n',y_test[:10])`

Explanation:
Predictions are made on the test dataset using the trained model. The predicted values are printed alongside the actual values for a quick comparison.

---

7. Confusion Matrix and Display

- **Head:** `matrix = confusion_matrix(...)`
- **Tail:** `plt.show()`

Explanation:
The confusion matrix is computed to evaluate model performance in terms of TP, TN, FP, FN. It is displayed visually using `ConfusionMatrixDisplay` to help interpret results.

---

8. Classification Report and Metrics

- **Head:** `print(classification_report(y_test,pred))`
- **Tail:** `print('Precision (False Positive Rate):',fp/(tn+fp))`

Explanation:
A classification report is printed, summarizing precision, recall, F1-score, and support. Then, individual metrics are calculated manually from the confusion matrix to provide detailed insights into model performance like Accuracy, Error Rate, Recall, Specificity, Precision, and False Positive Rate.