**Cell 1: Import Libraries**

python

Copy

Download

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, confusion\_matrix

**Problem Step:**  
"Import all required Python libraries for logistic regression."

**Definitions:**

* pandas: Used for data manipulation (e.g., loading CSV files, handling DataFrames).
* train\_test\_split: Splits data into training and testing sets for model evaluation.
* LogisticRegression: Implements the logistic regression algorithm for classification.
* accuracy\_score, confusion\_matrix: Metrics to evaluate model performance.

**Why:**

* Pandas loads and preprocesses the dataset.
* train\_test\_split ensures the model is tested on unseen data.
* LogisticRegression fits the model to predict binary outcomes (e.g., purchased or not).
* Metrics quantify prediction accuracy and classification performance.

**Cell 2: Load Data**

python

Copy

Download

df = pd.read\_csv('datasets/Social\_Network\_Ads.csv')

**Problem Step:**  
"Load the dataset from a CSV file."

**Definition:**

* pd.read\_csv(): Reads a CSV file into a pandas DataFrame.

**Why:**  
The dataset contains features (Age, EstimatedSalary, Gender) and the target (Purchased) for classification.

**Cell 3: Inspect Data**

python

Copy

Download

df.head()

**Problem Step:**  
"Check the first few rows of the dataset."

**Definition:**

* df.head(): Displays the first 5 rows of the DataFrame.

**Why:**  
Quickly verify the data structure (e.g., column names, values) to ensure proper loading.

**Cell 4: Data Info**

python

Copy

Download

df.info()

**Problem Step:**  
"Summarize the dataset (e.g., data types, missing values)."

**Definition:**

* df.info(): Prints column names, non-null counts, and data types.

**Why:**  
Confirms there are no missing values and that data types are correct for analysis.

**Cell 5: Encode Categorical Data**

python

Copy

Download

mapping = {'Male':0, 'Female':1}

df['Gender'] = df['Gender'].map(mapping)

**Problem Step:**  
"Convert categorical data (Gender) to numerical values."

**Definition:**

* map(): Replaces categorical values with numerical equivalents (e.g., Male → 0, Female → 1).

**Why:**  
Logistic regression requires numerical input features.

**Cell 6: Verify Encoding**

python

Copy

Download

df.head()

**Problem Step:**  
"Confirm the encoding was applied correctly."

**Why:**  
Ensures the Gender column is now numerical (0/1).

**Cell 7: Split Features and Target**

python

Copy

Download

X = df.iloc[:,1:-1]

y = df.iloc[:,-1]

**Problem Step:**  
"Separate features (X) and target (y)."

**Definition:**

* iloc: Selects columns by position (features: columns 1 to -2; target: last column).

**Why:**  
Prepares data for model training by isolating input features and the output variable.

**Cell 8: Split Data into Train/Test Sets**

python

Copy

Download

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

**Problem Step:**  
"Divide data into training (80%) and testing (20%) sets."

**Definition:**

* train\_test\_split: Randomly splits data while preserving class distribution.

**Why:**  
Ensures the model is evaluated on unseen data (test\_size=0.2).  
random\_state=42 ensures reproducibility.

**Cell 9: Feature Scaling**

python

Copy

Download

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

**Problem Step:**  
"Standardize features to have zero mean and unit variance."

**Definitions:**

* StandardScaler: Normalizes features to improve model performance.
* fit\_transform: Computes mean/variance and scales training data.
* transform: Applies the same scaling to test data.

**Why:**  
Logistic regression is sensitive to feature scales; standardization ensures fair weighting.

**Cell 10: Train Logistic Regression Model**

python

Copy

Download

model = LogisticRegression()

model.fit(X\_train, y\_train)

**Problem Step:**  
"Train a logistic regression model on the training data."

**Definitions:**

* LogisticRegression(): Initializes the classifier.
* fit(): Trains the model using the training data.

**Why:**  
Fits a decision boundary to predict binary outcomes (Purchased: 0 or 1).

**Cell 11: Make Predictions**

python

Copy

Download

y\_pred = model.predict(X\_test)

**Problem Step:**  
"Predict class labels for the test set."

**Definition:**

* predict(): Generates predictions (0 or 1) for X\_test.

**Why:**  
Evaluates how well the model generalizes to new data.

**Cell 12: Calculate Accuracy**

python

Copy

Download

accuracy\_score(y\_test, y\_pred)

**Problem Step:**  
"Measure overall prediction accuracy."

**Definition:**

* accuracy\_score: Computes the proportion of correct predictions.

**Why:**  
Accuracy of 0.8875 means 88.75% of predictions are correct.

**Cell 13: Compute Confusion Matrix**

python

Copy

Download

cm = confusion\_matrix(y\_test, y\_pred)

cm

**Problem Step:**  
"Generate a confusion matrix to evaluate classification performance."

**Definition:**

* confusion\_matrix: Tabulates true/false positives/negatives.

**Why:**  
Provides a breakdown of prediction errors for deeper analysis.

**Cell 14: Extract TP, FP, TN, FN**

python

Copy

Download

TN, FP, FN, TP = cm.ravel()

**Problem Step:**  
"Extract values from the confusion matrix."

**Definition:**

* ravel(): Flattens the confusion matrix into individual components.

**Why:**  
Enables calculation of metrics like precision, recall, and error rate.

**Cell 15: Print Confusion Matrix Values**

python

Copy

Download

print(f'TP: {TP}\tFP: {FP}\nTN: {TN}\tFN: {FN}')

**Problem Step:**  
"Display the confusion matrix components."

**Why:**  
Shows the counts of true/false predictions for clarity.

**Cell 16: Calculate Metrics**

python

Copy

Download

accuracy = (TP + TN)/(TP + TN + FP + FN)

precision = TP/(TP + FP)

recall = TP/(TP + FN)

f1\_score = 2 \* ((precision \* recall)/(precision + recall))

error\_rate = 1 - accuracy

**Problem Step:**  
"Compute accuracy, precision, recall, F1-score, and error rate."

**Definitions:**

* **Accuracy**: Overall correctness.
* **Precision**: Proportion of true positives among predicted positives.
* **Recall**: Proportion of true positives among actual positives.
* **F1-score**: Harmonic mean of precision and recall.
* **Error Rate**: Proportion of incorrect predictions.

**Why:**  
Quantifies different aspects of model performance for comprehensive evaluation.

**Cell 17: Print Metrics**

python

Copy

Download

print(f'Accuracy: {accuracy:.4f}\nPrecision: {precision:.4f}\nRecall: {recall:.4f}\nF1-score: {f1\_score:.4f}\nError Rate: {error\_rate:.4f}')

**Problem Step:**  
"Display the computed metrics."

**Why:**  
Summarizes model performance in an interpretable format.

**Cell 18: Predict for New Data**

python

Copy

Download

inputs = pd.DataFrame([[0, 20.0, 61000.0]], columns=['Gender', 'Age', 'EstimatedSalary'])

**Problem Step:**  
"Prepare new data for prediction."

**Why:**  
Demonstrates how to use the trained model for real-world predictions.

**Cell 19: Scale and Predict**

python

Copy

Download

inputs = sc.transform(inputs)

model.predict(inputs)[0]

**Problem Step:**  
"Predict the class for new input."

**Why:**  
Shows the model's prediction (0 = not purchased) for a user with Age=20, EstimatedSalary=61000.