Name – Shivani kinagi

PRN – 22311443

Roll No – 282048

Year – SY B

**Assignment No – 4**

**Problem Statement :**

Apply appropriate ML algorithm on a dataset collected in a cosmetics shop showing details of

customers to predict customer response for special offer.

Create confusion matrix based on above data and find

a) Accuracy

b) Precision

c) Recall

d) F-1 score

**Objective**

By addressing its present shortcomings, the goal is to improve a simple and intelligible machine learning technique for forecasting consumer reactions using a dataset from a cosmetics store. By using data preprocessing (e.g., scaling, encoding), integrating visualizations (e.g., confusion matrix heatmaps, feature distributions), performing extensive data exploration (e.g., summary statistics, correlation analysis), and making sure the dataset is in line with the problem context—thereby reducing overfitting and data mismatch issues—the objective is to improve accuracy and relevance. This will turn the approach from a sound foundation into a reliable, workable solution appropriate for actual use.

**Methodology**

The code follows a standard supervised machine learning workflow to predict customer response (though it uses heart.csv instead of a cosmetics dataset):

1. **Library Import**: Import numpy, pandas, and sklearn modules for data manipulation, modeling, and evaluation.
2. **Data Loading**: Load the heart.csv dataset and split it into features (X) and target (y).
3. **Data Splitting**: Divide the dataset into training (80%) and testing (20%) sets using train\_test\_split.
4. **Model Training**: Train a Logistic Regression classifier on the training data.
5. **Prediction**: Use the trained model to predict outcomes on the test set.
6. **Evaluation**: Generate a confusion matrix and calculate accuracy, precision, recall, and F1-score to assess model performance.

**Main Functions**

1. **numpy (np)**
   * **Used**: Imported but not explicitly utilized in the code.
   * **Purpose**: Provides numerical array support underlying pandas and sklearn.
2. **pandas (pd)**
   * **pd.read\_csv('heart.csv')**: Loads the dataset into a DataFrame.
   * **dataset.iloc[:, :-1].values**: Extracts feature columns as a NumPy array (X).
   * **dataset.iloc[:, -1].values**: Extracts the target column as a NumPy array (y).
3. **sklearn.model\_selection**
   * **train\_test\_split(X, y, test\_size=0.2, random\_state=0)**: Splits data into training and testing sets.
4. **sklearn.linear\_model**
   * **LogisticRegression()**: Initializes a logistic regression classifier.
   * **classifier.fit(X\_train, y\_train)**: Trains the model on the training data.
   * **classifier.predict(X\_test)**: Predicts outcomes for the test set.
5. **sklearn.metrics**
   * **confusion\_matrix(y\_test, y\_pred)**: Creates a confusion matrix.
   * **accuracy\_score(y\_test, y\_pred)**: Computes the accuracy of predictions.
   * **precision\_score(y\_test, y\_pred)**: Calculates precision (positive predictive value).
   * **recall\_score(y\_test, y\_pred)**: Calculates recall (sensitivity).
   * **f1\_score(y\_test, y\_pred)**: Computes the F1-score (harmonic mean of precision and recall).

**Advantages :**

* NumPy: Supports other libraries and performs array operations quickly.
* Pandas: Simple data loading, adaptable slicing, and sklearn integration.
* Scikit-learn (model\_selection): Easy data division that can be replicated using random\_state.  
  Scikit-learn (linear\_model): A good baseline model for logistic regression that is simple to use.
* Scikit-learn (metrics): Easy to use, comprehensive evaluation metrics.

**Disadvantages:**

* NumPy: Low-level for stand-alone tasks, not utilized here.
* Pandas: Memory-intensive for huge data, limited in use beyond loading.
* Scikit-learn (model\_selection): Stratification for imbalance is absent from basic splitting.
* Scikit-learn (linear\_model): Assumes linearity and has convergence problems without scaling.
* Scikit-learn measurements are susceptible to class imbalance and lack visualization.

**Conclusion**   
Using `pandas` to load and prepare the `heart.csv` dataset, `sklearn` to split the data, and a confusion matrix and metrics (accuracy: 0.86, precision: 0.83, recall: 0.93, F1-score: 0.88) to assess the model's performance, the Jupyter Notebook code implements a simple machine learning pipeline. Although it successfully illustrates a predictive workflow, the choice of a heart disease dataset makes it inconsistent with the claimed objective of forecasting consumer reactions in a cosmetics store. Though `numpy` is underutilized and the absence of preprocessing (e.g., scaling) results in a convergence warning in Logistic Regression, the libraries `numpy`, `pandas`, and `sklearn` offer strong functionality—`pandas` for data handling, `sklearn` for modeling and evaluation. The method is straightforward and easy to understand, but it is constrained by the lack of visualization, the scant data exploration, and the possibility of overfitting or data mismatch. Accuracy and relevance could be improved by preprocessing, displaying the results, and matching the dataset to the issue. It's a good starting point overall, but it has to be improved for real-world use.



