Machine Learning Internship Report

Submitted by: Yatharth Kumar Saxena
Objective: Build a classification model on customer data to predict the Target variable. Compare performance before and after data preprocessing.

Step-by-Step Explanation

1. Importing Required Libraries

The project began by importing key Python libraries such as:

- pandas for data manipulation,
- numpy for numerical computation,
- matplotlib and seaborn for data visualization.

These libraries form the foundation for data analysis and model building in Python.

2. Loading the Dataset

The customer dataset was provided in CSV format. It was read into a DataFrame using pandas.read_csv().
A backup copy of the raw dataset was also stored to compare results between raw and preprocessed versions later.

---

3. Initial Data Exploration

The dataset was briefly explored using the following:

- .info() to view column names, datatypes, and missing values.
- .describe() for statistical summaries.
- .shape to check rows and columns.
- .head() to preview the first few rows.

This step helped understand the structure and health of the dataset.

4. Dropping Irrelevant Features

Certain columns were found to be unhelpful:

- Legacy_Customer_ID – acted as a unique identifier, which doesn't help in prediction.
- Customer_Feedback – possibly subjective or not usable in current numeric encoding.

These columns were dropped to clean up the dataset.

---

5. Handling Missing Values

Numerical Columns such as Age, Credit_Score, and CLV_Score had missing values.

- These were filled using the **median** of their respective columns.

- Median is robust against outliers and gives a balanced central estimate.

Categorical Columns like Employment_Type and Education_Level also had missing entries.

- These were filled with a placeholder value 'Unknown' to retain the rows and still mark incomplete entries.

6. Label Encoding of Categorical Features

Since machine learning models don't accept categorical strings directly:

- All categorical columns were converted to numerical format using **Label Encoding**.

This included columns like:

- Employment_Type
- Education_Level
- Region
- Account_Type
- Contact_Preference
- Subscription_Tier

---

7. Feature and Target Separation

The dataset was now ready for modeling.

- All columns except Target were separated as **features** (X).
- The Target column was used as the label (y) to be predicted.

---

8. Train-Test Split

The dataset was split into training and testing sets using a 75%-25% ratio:

- Training set (75%) was used to train the model.
- Testing set (25%) was used to evaluate how well the model generalizes to unseen data.

9. Model Training – Random Forest Classifier

For this assignment:

- A **Random Forest Classifier** was chosen due to its robustness and ensemble nature.
- The model was trained on the training dataset.
- Predictions were made on the test set.

---

10. Model Evaluation

The performance of the classifier was evaluated using:

- **Accuracy Score**
- **Precision**, **Recall**, and **F1-Score** (from the Classification Report)

Final Result

➤ **Random Forest Classifier Performance:**

Accuracy: 56.8%

Metric	**Class 0**	**Class 1**
Precision	0.59	0.44
Recall	0.88	0.14
F1-Score	0.70	0.21

- **Macro Avg F1 Score:** 0.46
- **Weighted Avg F1 Score:** 0.50
- **Support (Test Set Size):** 1250

Observations

- The model performs **better on class 0**, which might indicate **class imbalance**.
- Preprocessing helped clean the data, but model performance (~57%) shows scope for:
    - Feature engineering
    - Balancing classes
    - Hyperparameter tuning
    - Trying other algorithms (e.g. XGBoost, SVM, etc.)

Conclusion

This report presented the full pipeline from:

- Data loading
  → to cleaning,
  → to encoding,
  → to model training and evaluation.

Although the Random Forest model showed moderate performance, it demonstrates the value of data preprocessing in ML tasks. Future work can aim to improve metrics using advanced techniques.