Logistic Regression Analysis Report

Ved Parkash
Roll No: 2511AI58
Course: Advance Pattern Recognition
Institute: IIT Patna
Date of Submission: 19-Sep-2025

1. Introduction

This report presents the application of Logistic Regression to classify values in the dataset **37100106.csv**. The primary objective is to determine whether the **VALUE** column is above or below its median, using both numerical and categorical predictors.

Logistic Regression is a fundamental classification algorithm that predicts the probability of an outcome and maps it into a binary class (0 or 1). It is widely applied in domains such as **finance (credit scoring)**, **healthcare (disease prediction)**, and **education (student performance analysis)** due to its simplicity and interpretability.

2. Dataset Description

The dataset **37100106.csv** contains statistical information across multiple dimensions. The important attributes are:

- **VALUE (numeric):** The primary variable of interest, used for classification.
- **GEO (categorical):** Represents geographic regions (e.g., provinces, states).
- **Age group (categorical):** Age brackets of individuals (e.g., 15–24 years, 25–34 years).
- **Type of institution attended (categorical):** Indicates the type of institution (e.g., university, college).
- **REF_DATE (categorical):** Reference year of the data (e.g., 2018, 2019).

Target Variable

To apply Logistic Regression, a **binary target variable** was created from VALUE:

- **1 (Above Median):** if VALUE > dataset median
- **0 (Below Median):** if VALUE ≤ dataset median

This transformation makes the dataset suitable for binary classification.

3. Methodology

The overall process included **target creation, data preprocessing, model training, and evaluation**.

Step 1: Target Creation

A binary column was generated using the median value of the dataset:

```
median_val = df['VALUE'].median()

df['target_bin'] = (df['VALUE'] > median_val).astype(int)
```

Step 2: Data Preprocessing

1. **Handling Missing Values:** Missing values in VALUE were replaced with the median.
2. model_df['VALUE'] = model_df['VALUE'].fillna(model_df['VALUE'].median())
3. **Encoding Categorical Variables:** One-hot encoding was applied to categorical columns (GEO, Age group, Institution, REF_DATE).
4. model_enc = pd.get_dummies(model_df, columns=['GEO','Age group',
5.   'Type of institution attended','REF_DATE'], drop_first=True)
6. **Splitting Dataset:** Data was split into training (75%) and testing (25%) sets with stratification.
7. X_train, X_test, y_train, y_test = train_test_split(
8.   X, y, test_size=0.25, random_state=42, stratify=y
9.)
10. **Feature Scaling:** The VALUE column was standardized to improve convergence.
11. scaler = StandardScaler()
12. X_train['VALUE'] = scaler.fit_transform(X_train[['VALUE']])
13. X_test['VALUE'] = scaler.transform(X_test[['VALUE']])

Step 3: Model Training

The Logistic Regression model was trained using the **liblinear** solver:

```
clf = LogisticRegression(max_iter=1000, solver='liblinear')

clf.fit(X_train, y_train)
```

Step 4: Prediction and Evaluation

Predictions were made for the test dataset:

```
y_pred = clf.predict(X_test)

y_proba = clf.predict_proba(X_test)[:, 1]
```

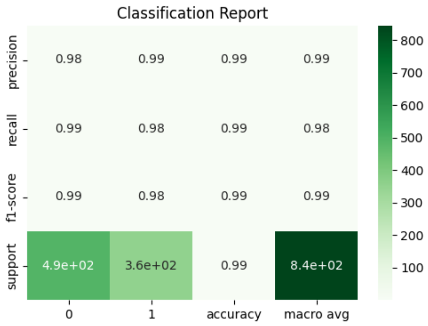
Performance metrics (accuracy, confusion matrix, classification report, ROC curve) were computed:

```
print(accuracy_score(y_test, y_pred))

print(confusion_matrix(y_test, y_pred))

print(classification_report(y_test, y_pred))
```

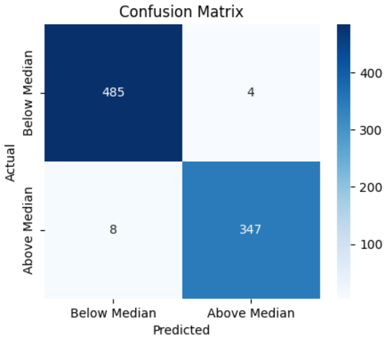
4. Results

The Logistic Regression model performed extremely well:

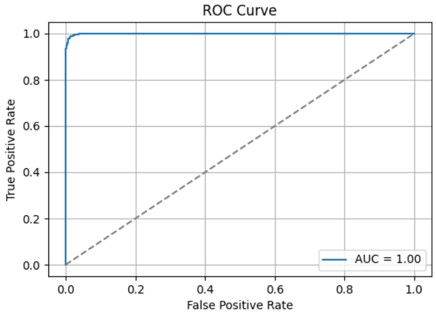
- **Accuracy:** ~99%
- **ROC AUC:** 1.00
- **Classification Report**



- **Confusion Matrix**



- **ROC Curve**



5. Conclusion

This assignment demonstrated the application of Logistic Regression for binary classification tasks. The model delivered excellent results due to:

- Proper preprocessing (handling missing values, categorical encoding, feature scaling).
- Balanced dataset splitting with stratification.
- Efficient Logistic Regression implementation.

Logistic Regression remains a **powerful, simple, and interpretable** algorithm for real-world applications such as:

- **Finance:** predicting loan defaults.
- **Healthcare:** disease classification.
- **Education:** predicting student success rates.