

Experiment - 1

Nikhil Kale
D15A/50

Aim - Implement Linear and Logistic Regression on a Real-World Dataset

1. Dataset Source

Dataset Name: Bank Marketing Dataset

Source: Kaggle

<https://www.kaggle.com/datasets/janiobachmann/bank-marketing-dataset>

This dataset contains information about bank customers and whether they subscribed to a term deposit after a marketing campaign.

2. Dataset Description

The dataset is related to direct marketing campaigns of a Portuguese banking institution.

Dataset Size

- ~11,000 records (bank.csv version)
- 17 input features
- 1 target variable

Target Variable

- **deposit** → (Yes/No)
 - Yes → Customer subscribed to term deposit
 - No → Customer did not subscribe

Important Features

- age (numeric)
- job (categorical)
- marital (categorical)
- education (categorical)
- balance (numeric)
- housing (yes/no)
- loan (yes/no)

- duration (numeric)
- campaign (numeric)
- outcome (categorical)

Data Characteristics

- Mixed numerical and categorical data
- No missing values
- Binary classification problem

3. Mathematical Formulation of the Algorithm

Linear Regression

Linear regression models the relationship between independent variables and a continuous dependent variable.

$$\hat{y} = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n$$

Where:

- β_0 = intercept
- β_i = coefficients
- X_i = features

Objective: Minimize Residual Sum of Squares (RSS):

$$RSS = \sum (y - \hat{y})^2$$

Logistic Regression

Used for binary classification.

$$P(Y = 1|X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \dots + \beta_n X_n)}}$$

Uses sigmoid function to convert linear output into probability.

Decision rule:

If $P \geq 0.5 \Rightarrow 1$, else 0
If $P \leq 0.5 \Rightarrow 1$, else 0

4. Algorithm Limitations

Linear Regression

- Assumes linear relationship
- Not suitable for binary target variables
- Sensitive to outliers

Logistic Regression

- Assumes linear decision boundary
- Cannot model complex nonlinear relationships
- Performance decreases with highly correlated features

5. Methodology / Workflow

Step 1: Data Loading

Loaded dataset using Pandas.

Step 2: Data Preprocessing

- Encoded categorical variables using One-Hot Encoding
- Converted target variable to 0 and 1
- Split dataset (80% training, 20% testing)
- Applied feature scaling using StandardScaler

Step 3: Model Training

- Trained Logistic Regression model
- Trained Linear Regression model (for academic comparison)

Step 4: Model Evaluation

- Accuracy
- Confusion Matrix
- Precision, Recall, F1-score
- R^2 (for linear regression)

Workflow Diagram

Dataset



Data Cleaning & Encoding



Train-Test Split



Feature Scaling



Model Training



Prediction



Evaluation

6. Performance Analysis

Logistic Regression Results

- Accuracy: **80.83%**
- Precision: ~81%
- Recall: ~78%
- F1 Score: ~80%

The model correctly classified approximately 81% of customers.

Recall for deposit subscribers was slightly lower, indicating some potential customers were missed.

Linear Regression Result

- R^2 Score: Low (as expected since target is binary)

Logistic Regression performed significantly better for classification.

7. Hyperparameter Tuning

GridSearchCV was used to tune parameter:

- C (inverse of regularization strength)

Example values tested:

- 0.01
- 0.1
- 1
- 10
- 100

Impact:

- Slight improvement in recall
- Better model stability
- Reduced overfitting risk

Code and Output -

```
[1] 2s
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, r2_score

[2] 0s
df = pd.read_csv('/content/bank.csv')
df.head()
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	deposit
0	59	admin.	married	secondary	no	2343	yes	no	unknown	5	may	1042	1	-1	0	unknown	yes
1	56	admin.	married	secondary	no	45	no	no	unknown	5	may	1467	1	-1	0	unknown	yes
2	41	technician	married	secondary	no	1270	yes	no	unknown	5	may	1389	1	-1	0	unknown	yes
3	55	services	married	secondary	no	2476	yes	no	unknown	5	may	579	1	-1	0	unknown	yes
4	54	admin.	married	tertiary	no	184	no	no	unknown	5	may	673	2	-1	0	unknown	yes

```
[4]    ✓ 0s      le = LabelEncoder()

          for column in df.select_dtypes(include='object').columns:
              df[column] = le.fit_transform(df[column])

[13]    ✓ 0s      X = df.drop('deposit', axis=1)
              y = df['deposit']

[16]    ✓ 0s      y = y.map({'yes': 1, 'no': 0})

[17]    ✓ 0s      X = pd.get_dummies(X, drop_first=True)

[18]    ✓ 0s      X_train, X_test, y_train, y_test = train_test_split(
              X, y, test_size=0.2, random_state=42
            )

[19]    ✓ 0s      scaler = StandardScaler()

              X_train = scaler.fit_transform(X_train)
              X_test = scaler.transform(X_test)
```

```
[20] ✓ 0s ⏪ log_model = LogisticRegression(max_iter=1000)
log_model.fit(X_train, y_train)

y_pred = log_model.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))

▼ ... Accuracy: 0.8083296014330497

Confusion Matrix:
[[972 194]
 [234 833]]

Classification Report:
              precision    recall   f1-score   support
          0       0.81      0.83      0.82     1166
          1       0.81      0.78      0.80     1067

      accuracy           0.81      2233
     macro avg       0.81      0.81      0.81     2233
  weighted avg       0.81      0.81      0.81     2233
```

Linear Regression -

```
[21] ✓ 0s ⏪ lin_model = LinearRegression()
lin_model.fit(X_train, y_train)

y_pred_lin = lin_model.predict(X_test)

print("R2 Score:", r2_score(y_test, y_pred_lin))

▼ ... R2 Score: 0.4116151112510271
```