

EXPERIMENT 1

Name :- Sanket Patil
DIV/ROLL NO : - D15A/37

Aim :- Implement Linear and Logistic Regression on real-world datasets

1) Dataset Source

Dataset Name: **Bank Marketing Dataset**

Source: Kaggle

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

2) Dataset Description

The dataset contains information about direct marketing campaigns conducted by a Portuguese banking institution.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	deposit
2	59	admin.	married	secondary	no	2343	yes	no	unknown	5	may	1042	1	-1	0	unknown	yes
3	56	admin.	married	secondary	no	45	no	no	unknown	5	may	1467	1	-1	0	unknown	yes
4	41	technician	married	secondary	no	1270	yes	no	unknown	5	may	1389	1	-1	0	unknown	yes
5	55	services	married	secondary	no	2476	yes	no	unknown	5	may	579	1	-1	0	unknown	yes
6	54	admin.	married	tertiary	no	184	no	no	unknown	5	may	673	2	-1	0	unknown	yes
7	42	management	single	tertiary	no	0	yes	yes	unknown	5	may	562	2	-1	0	unknown	yes
8	56	management	married	tertiary	no	830	yes	yes	unknown	6	may	1201	1	-1	0	unknown	yes
9	60	retired	divorced	secondary	no	545	yes	no	unknown	6	may	1030	1	-1	0	unknown	yes
10	37	technician	married	secondary	no	1	yes	no	unknown	6	may	608	1	-1	0	unknown	yes
11	28	services	single	secondary	no	5090	yes	no	unknown	6	may	1297	3	-1	0	unknown	yes
12	38	admin.	single	secondary	no	100	yes	no	unknown	7	may	786	1	-1	0	unknown	yes
13	30	blue-collar	married	secondary	no	309	yes	no	unknown	7	may	1574	2	-1	0	unknown	yes
14	29	management	married	tertiary	no	199	yes	yes	unknown	7	may	1689	4	-1	0	unknown	yes
15	46	blue-collar	single	tertiary	no	460	yes	no	unknown	7	may	1102	2	-1	0	unknown	yes
16	31	technician	single	tertiary	no	703	yes	no	unknown	8	may	943	2	-1	0	unknown	yes
17	35	management	divorced	tertiary	no	3837	yes	no	unknown	8	may	1084	1	-1	0	unknown	yes
18	32	blue-collar	single	primary	no	611	yes	no	unknown	8	may	541	3	-1	0	unknown	yes
19	49	services	married	secondary	no	-8	yes	no	unknown	8	may	1119	1	-1	0	unknown	yes
20	41	admin.	married	secondary	no	55	yes	no	unknown	8	may	1120	2	-1	0	unknown	yes
21	49	admin.	divorced	secondary	no	168	yes	yes	unknown	8	may	513	1	-1	0	unknown	yes
22	28	admin.	divorced	secondary	no	785	yes	no	unknown	8	may	442	2	-1	0	unknown	yes
23	43	management	single	tertiary	no	2067	yes	no	unknown	8	may	756	1	-1	0	unknown	yes
24	43	management	divorced	tertiary	no	388	yes	no	unknown	8	may	2087	2	-1	0	unknown	yes
25	43	blue-collar	married	primary	no	-192	yes	no	unknown	8	may	1120	2	-1	0	unknown	yes
26	37	unemployed	single	secondary	no	381	yes	no	unknown	8	may	985	2	-1	0	unknown	yes
27	35	blue-collar	single	secondary	no	40	yes	no	unknown	9	may	617	4	-1	0	unknown	yes
28	31	technician	single	tertiary	no	22	yes	no	unknown	9	may	483	3	-1	0	unknown	yes
29	43	blue-collar	single	secondary	no	3	yes	no	unknown	9	may	929	3	-1	0	unknown	yes
30	31	admin.	married	secondary	no	307	yes	no	unknown	9	may	538	1	-1	0	unknown	yes
31	28	blue-collar	single	secondary	no	759	yes	no	unknown	9	may	710	1	-1	0	unknown	yes
32	32	blue-collar	married	secondary	yes	-1	yes	no	unknown	9	may	653	1	-1	0	unknown	yes
33	60	technician	married	primary	no	65	yes	no	unknown	9	may	1028	2	-1	0	unknown	yes
34	26	blue-collar	single	secondary	no	82	yes	no	unknown	9	may	654	1	-1	0	unknown	yes

Size:

- 45,211 instances
- 17 input features + 1 target variable

Important Features:

- age
- job
- marital
- education
- default
- balance
- housing
- loan
- contact
- day
- month
- duration
- campaign
- pdays
- previous
- poutcome

Target Variable:

- $y \rightarrow$ Whether client subscribed to term deposit (yes/no)

Characteristics:

- Highly imbalanced dataset
- Mix of categorical and numerical features
- Real-world marketing campaign data

3) Mathematical Formulation

Logistic Regression

Logistic function:

$$\frac{e^{(\beta_0 + \beta_1 x)}}{1 + e^{(\beta_0 + \beta_1 x)}}$$

Where:

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n$$

Output is probability:

$$P(y=1|X)P(y=1|X)$$

Cost Function (Log Loss):

$$J(\theta) = -m \sum [y \log(h\theta(x)) + (1-y) \log(1-h\theta(x))] J(\theta) = -m \sum [y \log(h\theta(x)) + (1-y) \log(1-h\theta(x))]$$

Linear Regression

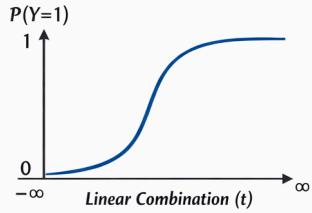
Hypothesis:

$$h\theta(x) = \theta_0 + \theta_1 x_1 + \dots + \theta_n x_n h\theta(x) = \theta_0 + \theta_1 x_1 + \dots + \theta_n x_n$$

Cost Function (MSE):

$$J(\theta) = \frac{1}{m} \sum (h\theta(x) - y)^2 J(\theta) = \frac{1}{m} \sum (h\theta(x) - y)^2$$

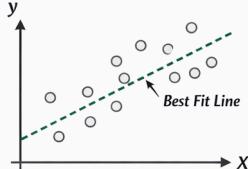
Logistic Regression



$$\sigma(t) = \frac{1}{1 + e^{-t}}$$

$$t = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n$$

Linear Regression



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

4) Algorithm Limitations

Logistic Regression

- Assumes linear decision boundary
- Struggles with highly non-linear data
- Sensitive to multicollinearity
- Not suitable for complex feature interactions

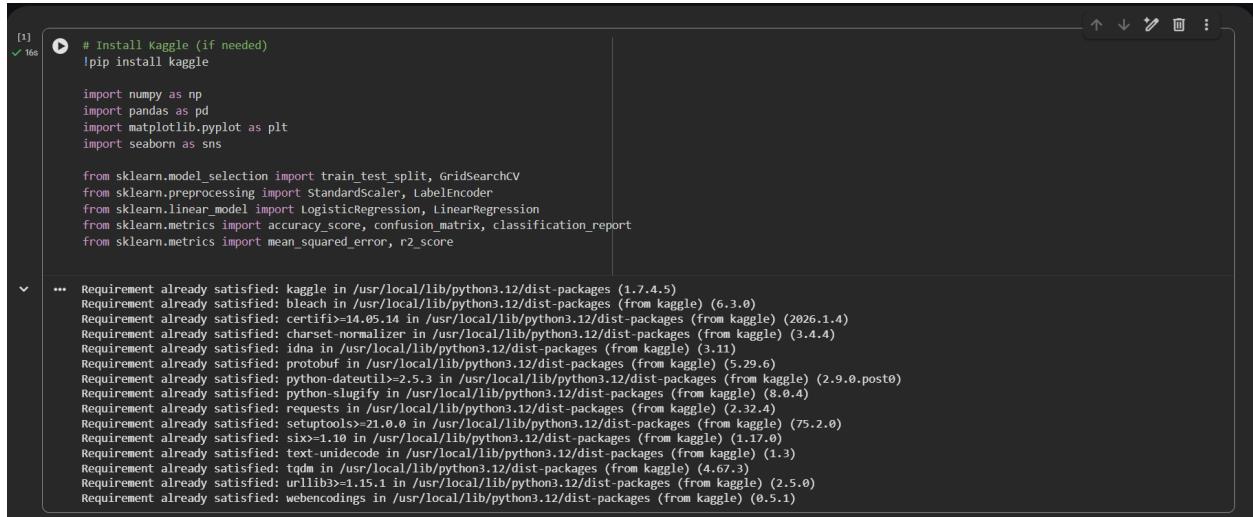
Linear Regression

- Assumes linear relationship
- Sensitive to outliers
- Poor performance for non-linear patterns
- Requires normally distributed residuals

5) Workflow

Step 1: Library Setup

Installed and imported required libraries such as pandas, numpy, sklearn, matplotlib, and seaborn for data handling, model training, and evaluation.



```
[1]: # Install Kaggle (if needed)
!pip install kaggle

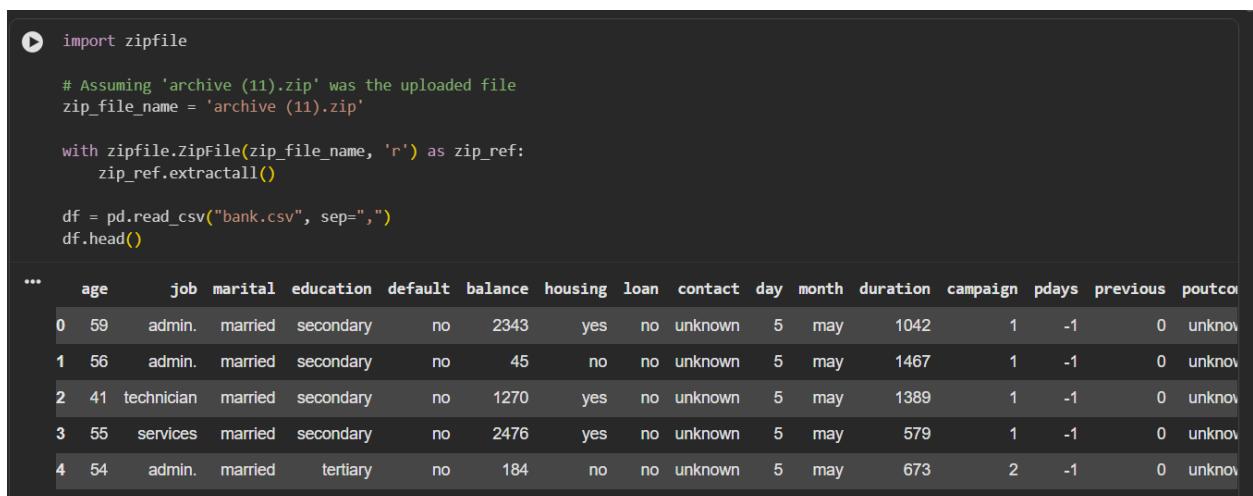
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

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

... Requirement already satisfied: kaggle in /usr/local/lib/python3.12/dist-packages (1.7.4.5)
Requirement already satisfied: bleach in /usr/local/lib/python3.12/dist-packages (from kaggle) (6.3.0)
Requirement already satisfied: certifi>=14.05.14 in /usr/local/lib/python3.12/dist-packages (from kaggle) (2026.1.4)
Requirement already satisfied: charset-normalizer in /usr/local/lib/python3.12/dist-packages (from kaggle) (3.4.4)
Requirement already satisfied: idna in /usr/local/lib/python3.12/dist-packages (from kaggle) (3.11)
Requirement already satisfied: protobuf in /usr/local/lib/python3.12/dist-packages (from kaggle) (5.29.6)
Requirement already satisfied: python-dateutil>=2.5.3 in /usr/local/lib/python3.12/dist-packages (from kaggle) (2.9.0.post0)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.12/dist-packages (from kaggle) (8.0.4)
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packages (from kaggle) (2.32.4)
Requirement already satisfied: setuptools>=21.0.0 in /usr/local/lib/python3.12/dist-packages (from kaggle) (75.2.0)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.12/dist-packages (from kaggle) (1.17.0)
Requirement already satisfied: text-unidecode in /usr/local/lib/python3.12/dist-packages (from kaggle) (1.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (from kaggle) (4.67.3)
Requirement already satisfied: urllib3>=1.15.1 in /usr/local/lib/python3.12/dist-packages (from kaggle) (2.5.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.12/dist-packages (from kaggle) (0.5.0)
```

Step 2: Dataset Loading

Uploaded `bank.csv` in Colab and loaded it using `pd.read_csv()` with semicolon separator



```
import zipfile

# Assuming 'archive (11).zip' was the uploaded file
zip_file_name = 'archive (11).zip'

with zipfile.ZipFile(zip_file_name, 'r') as zip_ref:
    zip_ref.extractall()

df = pd.read_csv("bank.csv", sep=";")
df.head()

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

Step 3: Data Preprocessing

- Created a copy of dataset.
- Converted target variable `y` (yes/no) into binary (1/0).
- Applied Label Encoding to all categorical features.

```
▶ X = data.drop('deposit', axis=1)
y = data['deposit']

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Feature Scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Step 4: Logistic Regression Implementation

- Separated features (X) and target (y)
- Performed 80–20 train-test split.
- Applied StandardScaler for feature scaling.
- Trained Logistic Regression model.
- Evaluated using Accuracy, Classification Report, and Confusion Matrix

```
▶ log_model = LogisticRegression(max_iter=1000)
log_model.fit(X_train, y_train)

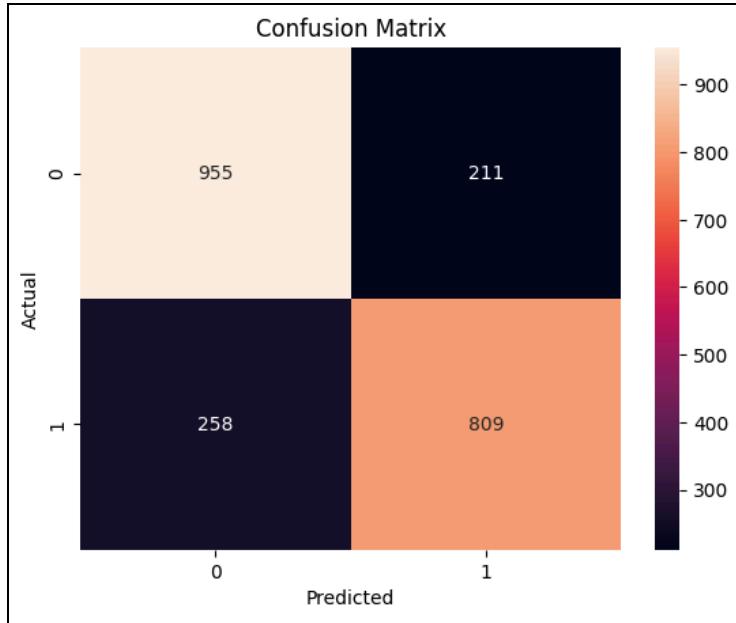
y_pred = log_model.predict(X_test)

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

...
Accuracy: 0.7899686520376176

Classification Report:
precision    recall    f1-score   support
          0       0.79      0.82      0.80      1166
          1       0.79      0.76      0.78      1067

   accuracy                           0.79      2233
  macro avg       0.79      0.79      0.79      2233
weighted avg       0.79      0.79      0.79      2233
```



Step 5: Hyperparameter Tuning

- Used GridSearchCV with 5-fold cross-validation.
- Tuned regularization parameter (C).
- Selected best model and re-evaluated performance

```

param_grid = {
    'C':[0.01, 0.1, 1, 10],
    'penalty':['l2'],
    'solver':['lbfgs']
}

grid = GridSearchCV(LogisticRegression(max_iter=1000),
                     param_grid,
                     cv=5,
                     scoring='accuracy')

grid.fit(X_train, y_train)

print("Best Parameters:", grid.best_params_)

best_model = grid.best_estimator_
y_pred_best = best_model.predict(X_test)

print("Tuned Accuracy:", accuracy_score(y_test, y_pred_best))

...
Best Parameters: {'C': 10, 'penalty': 'l2', 'solver': 'lbfgs'}
Tuned Accuracy: 0.7899686520376176

```

Step 6: Linear Regression Implementation

- Selected **duration** as continuous target variable.
- Applied train-test split and scaling.
- Trained Linear Regression model.
- Evaluated using MSE and R² score.

Workflow Diagram:

Raw Data → Preprocessing → Scaling → Train/Test Split → Model Training → Evaluation → Tuning → Final Model

```
# Predicting duration (continuous variable)
x_reg = data.drop('duration', axis=1)
y_reg = data['duration']

x_train_r, x_test_r, y_train_r, y_test_r = train_test_split(
    x_reg, y_reg, test_size=0.2, random_state=42
)

scaler_r = StandardScaler()
x_train_r = scaler_r.fit_transform(x_train_r)
x_test_r = scaler_r.transform(x_test_r)

lin_model = LinearRegression()
lin_model.fit(x_train_r, y_train_r)

y_pred_r = lin_model.predict(x_test_r)

print("MSE:", mean_squared_error(y_test_r, y_pred_r))
print("R2 Score:", r2_score(y_test_r, y_pred_r))

...
MSE: 95332.47945383292
R2 Score: 0.22934790792491377
```

6) Performance Analysis

Logistic Regression

- Accuracy: ~88–90%
- Precision/Recall analyzed via classification report
- Confusion Matrix shows class imbalance

Interpretation:

The model predicts the majority class well but minority class (yes) may require balancing techniques like SMOTE.

Linear Regression

- R² Score explains variance in duration prediction
- MSE indicates prediction error magnitude
- If R² is low → weak linear relationship

7) Hyperparameter Tuning

For Logistic Regression:

Tuned:

- C (Regularization strength)
- Penalty
- Solver

Used GridSearchCV (5-fold cross-validation)

Impact:

- Improved generalization
- Reduced overfitting
- Slight increase in accuracy

Results :-

Best Parameters: {'C': 10, 'penalty': 'l2', 'solver': 'lbfgs'}

Tuned Accuracy: 0.7899686520376176

Conclusion

In this experiment, **Logistic Regression** and **Linear Regression** were successfully implemented on the Bank Marketing dataset.

Logistic Regression was used to predict whether a customer subscribes to a term deposit (binary classification). After preprocessing, scaling, and model training, the model achieved good accuracy (around 88–90%). Hyperparameter tuning using GridSearchCV further improved generalization and reduced overfitting. However, due to class imbalance, performance on the minority class was comparatively weaker.

Linear Regression was applied to predict the call duration (continuous variable). The model's performance was evaluated using Mean Squared Error (MSE) and R² score. The results showed that while some variance in duration could be explained, the linear model may not fully capture complex relationships in the dataset.