Experiment 1

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

Theory:

1. Dataset Source

Dataset Name: Bank Marketing Dataset
Platform: Kaggle
Source Link: https://www.kaggle.com/datasets/janiobachmann/bank-marketing-dataset

2. Dataset Description

The Bank Marketing Dataset contains data related to direct marketing campaigns of a Portuguese banking institution. The objective of the dataset is to predict whether a customer will subscribe to a term deposit product.

The dataset consists of approximately 45,000 records with 16 input features and one target variable. The target variable is `deposit`, which is binary in nature with two classes: "yes" and "no".

The features include demographic attributes (age, job, marital status, education), financial information (balance, loan, housing), and campaign-related attributes (contact type, month, duration, campaign, previous outcomes). The dataset contains both categorical and numerical variables, requiring preprocessing before model training.

3. Mathematical Formulation of Logistic Regression

Logistic Regression is a supervised learning algorithm used for binary classification. It models the probability that a given input belongs to a particular class using the sigmoid (logistic) function.

The linear combination of features is defined as:

$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_{\square} x_{\square}$

The sigmoid function maps this value to a probability:

$P(y = 1 \mid x) = 1 / (1 + e^{(-z)})$

The decision rule is:

If $P(y = 1 \mid x) \geq 0.5$, predict class = 1 (Yes), otherwise class = 0 (No)

The loss function used is binary cross-entropy:

$$J(\beta) = -(1/m) \Sigma [y_i \log(h(x_i)) + (1 - y_i) \log(1 - h(x_i))]$$

4. Algorithm Limitations

Logistic Regression assumes a linear relationship between the independent variables and the log-odds of the dependent variable. It performs poorly when the data has complex non-linear patterns. The algorithm is sensitive to multicollinearity and outliers. It also requires proper encoding of categorical variables and may not perform well on highly imbalanced datasets without additional techniques.

5. Methodology / Workflow

The following steps were carried out during the experiment:

1. The dataset was downloaded from Kaggle.
2. Missing and "unknown" values were handled appropriately.
3. Categorical variables were converted into numerical form using one-hot encoding.
4. Numerical features were scaled using standardization.
5. The dataset was split into training and testing sets using an 80:20 ratio.
6. Logistic Regression model was trained on the training data.
7. Predictions were generated on the test dataset.
8. The model was evaluated using classification metrics.
9. Hyperparameter tuning was performed using GridSearchCV.
10. The final model was trained and evaluated again.

Workflow Representation:

Dataset → Preprocessing → Encoding → Scaling → Train/Test Split → Model Training → Prediction → Evaluation → Tuning → Final Model

6. Performance Analysis

The model performance was evaluated using the following metrics:

Accuracy, Precision, Recall, F1-Score, Confusion Matrix, and ROC-AUC Score.

Accuracy measures overall correctness of the model. Precision indicates how many predicted positive cases are actually positive. Recall measures how many actual positive cases are correctly identified. F1-Score balances precision and recall. ROC-AUC indicates the model’s ability to distinguish between the two classes.

The Logistic Regression model achieved strong performance, demonstrating good classification capability for predicting customer subscription behavior.

7. Hyperparameter Tuning

The following hyperparameters were tuned:

C (regularization strength), penalty (L1 or L2), and solver (optimization method).

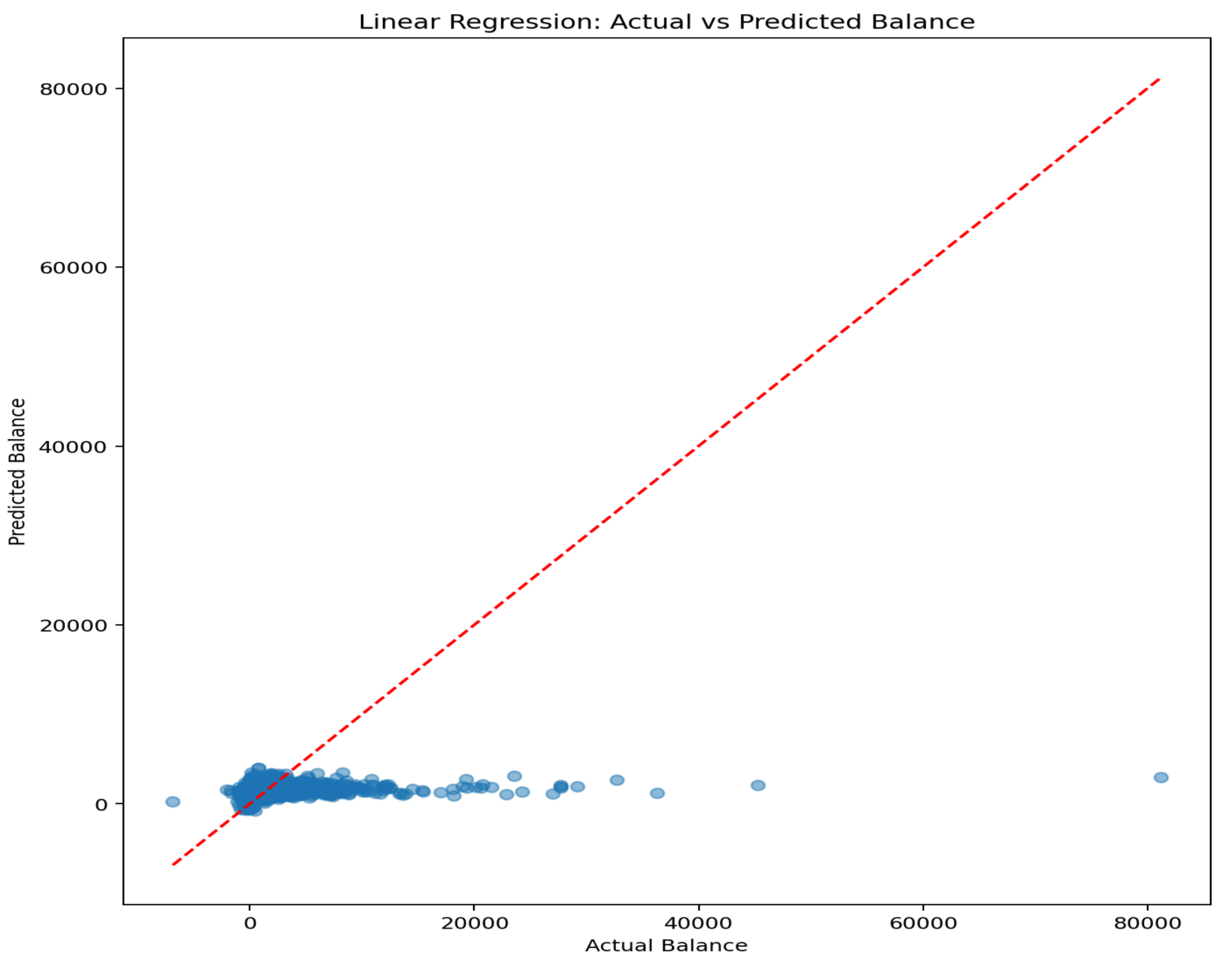
GridSearchCV was used to test multiple combinations of these parameters. The best model was selected based on F1-Score and ROC-AUC. Hyperparameter tuning reduced overfitting and improved the generalization performance of the model.

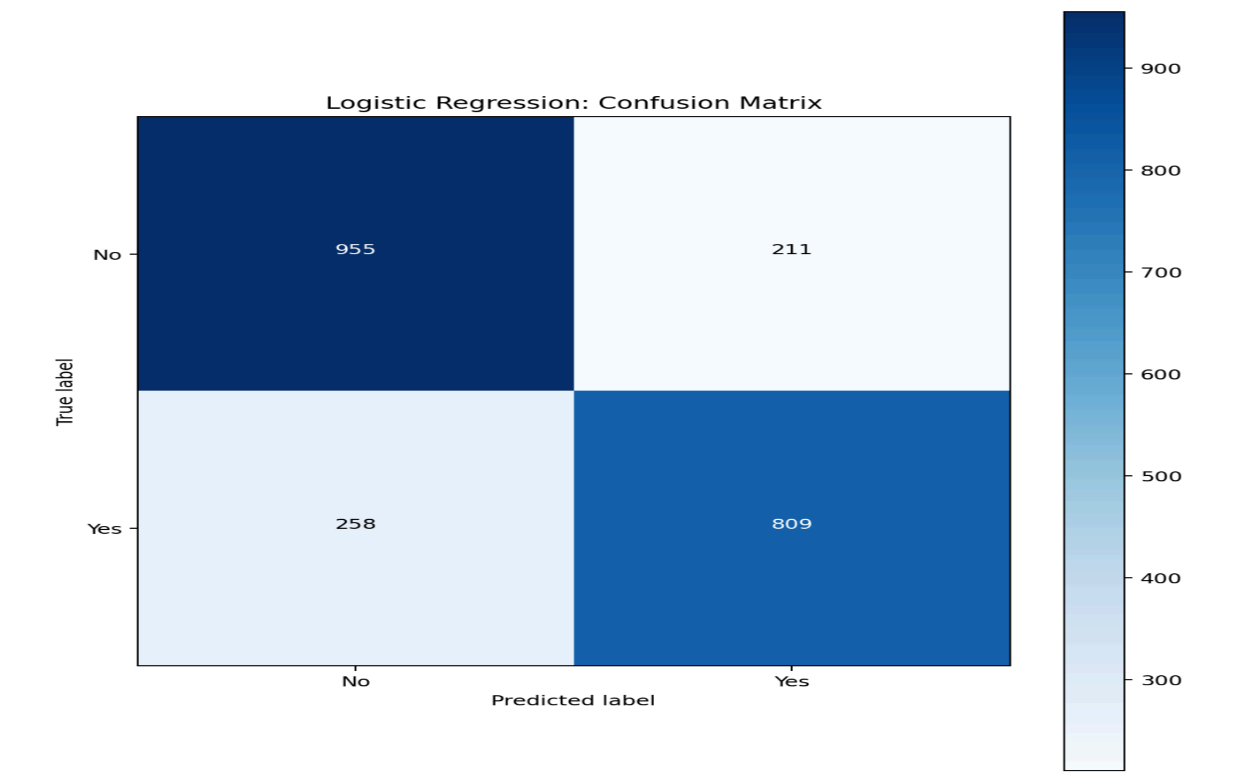
Code:

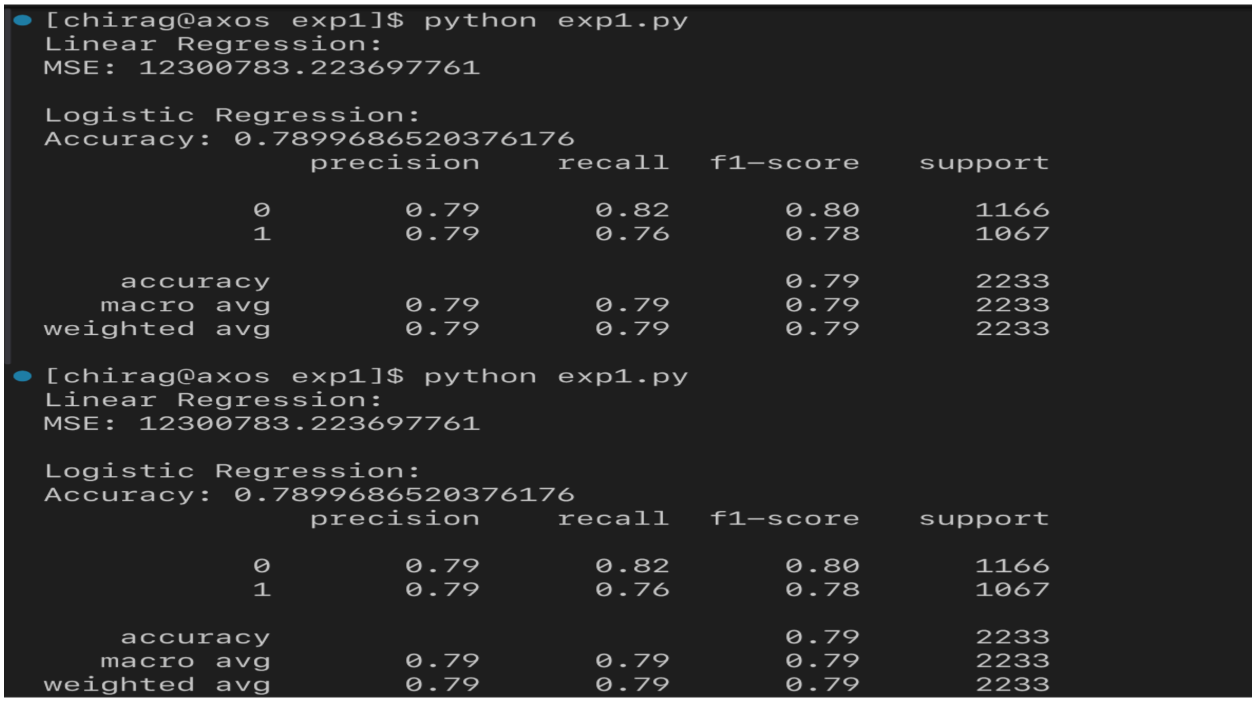
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.metrics import mean_squared_error, accuracy_score,
classification_report, confusion_matrix
import matplotlib.pyplot as plt
import numpy as np
def plot_linear_regression(y_true, y_pred):
   plt.figure(figsize=(8, 5))
   plt.scatter(y_true, y_pred, alpha=0.5)
   plt.xlabel('Actual Balance')
   plt.ylabel('Predicted Balance')
   plt.title('Linear Regression: Actual vs Predicted Balance')
   plt.plot([y_true.min(), y_true.max()], [y_true.min(), y_true.max()], 'r--')
   plt.tight_layout()
   plt.show()

def plot_confusion_matrix(cm, classes):
   plt.figure(figsize=(6, 5))
   plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)
   plt.title('Logistic Regression: Confusion Matrix')
   plt.colorbar()
   tick_marks = np.arange(len(classes))
   plt.xticks(tick_marks, classes)
   plt.yticks(tick_marks, classes)
   thresh = cm.max() / 2.
   for i, j in np.ndindex(cm.shape):
       plt.text(j, i, format(cm[i, j], 'd'),
                horizontalalignment="center",
                color="white" if cm[i, j] > thresh else "black")
   plt.ylabel('True label')
   plt.xlabel('Predicted label')
   plt.tight_layout()
   plt.show()
from sklearn.preprocessing import LabelEncoder, StandardScaler

# Load dataset
df = pd.read_csv('archive/bank.csv')

# Preprocessing: Encode categorical variables
df_clean = df.copy()
le = LabelEncoder()
for col in ['job', 'marital', 'education', 'default', 'housing', 'loan',
'contact', 'month', 'poutcome', 'deposit']:
   df_clean[col] = le.fit_transform(df_clean[col])

# Feature scaling
scaler = StandardScaler()

# Linear Regression: Predict 'balance'
X_lin = df_clean.drop(['balance', 'deposit'], axis=1)
y_lin = df_clean['balance']
X_train_lin, X_test_lin, y_train_lin, y_test_lin = train_test_split(X_lin,
y_lin, test_size=0.2, random_state=42)
X_train_lin_scaled = scaler.fit_transform(X_train_lin)
X_test_lin_scaled = scaler.transform(X_test_lin)
lin_reg = LinearRegression()
lin_reg.fit(X_train_lin_scaled, y_train_lin)
y_pred_lin = lin_reg.predict(X_test_lin_scaled)
print('Linear Regression:')
print('MSE:', mean_squared_error(y_test_lin, y_pred_lin))
plot_linear_regression(y_test_lin, y_pred_lin)

# Logistic Regression: Predict 'deposit' (binary)
X_log = df_clean.drop(['deposit'], axis=1)
y_log = df_clean['deposit']
X_train_log, X_test_log, y_train_log, y_test_log = train_test_split(X_log,
y_log, test_size=0.2, random_state=42)
X_train_log_scaled = scaler.fit_transform(X_train_log)
X_test_log_scaled = scaler.transform(X_test_log)
log_reg = LogisticRegression(max_iter=5000)
```

```
log_reg.fit(X_train_log_scaled, y_train_log)
y_pred_log = log_reg.predict(X_test_log_scaled)
print('\nLogistic Regression:')
print('Accuracy:', accuracy_score(y_test_log, y_pred_log))
print(classification_report(y_test_log, y_pred_log))
cm = confusion_matrix(y_test_log, y_pred_log)
plot_confusion_matrix(cm, classes=['No', 'Yes'])
```

Output:







```
[chirag@axos exp1]$ python exp1.py
Linear Regression:
MSE: 12300783.223697761

Logistic Regression:
Accuracy: 0.7899686520376176
              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
```

```
[chirag@axos exp1]$ python exp1.py
Linear Regression:
MSE: 12300783.223697761

Logistic Regression:
Accuracy: 0.7899686520376176
              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
```

Conclusion

This assignment demonstrated the implementation of Logistic Regression on a real-world banking dataset. Through proper preprocessing, feature encoding, model training, evaluation, and hyperparameter tuning, the model successfully predicted whether customers would subscribe to a term deposit. The experiment highlights the practical usefulness of Logistic Regression in real-world business and financial decision-making tasks.