D15A / 25

EXPERIMENT 1

AIM:

To implement Linear and Logistic Regression on real-world datasets

THEORY:

a. Linear Regression

1. Dataset Source

Dataset Name: Medical Cost Personal Dataset
Source: Kaggle
Link: https://www.kaggle.com/datasets/mirichoi0218/insurance

This dataset was used exclusively for this experiment.

2. Dataset Description

The Medical Insurance dataset contains information about individuals and their corresponding medical insurance charges.

Dataset Size:

- Total records: **1338**
- Total features: **6 input features + 1 target variable**

Features:

Feature	Description	Type
age	Age of the person	Numerical
sex	Gender (male/female)	Categorical
bmi	Body Mass Index	Numerical
children	Number of children	Numerical
smoker	Smoking status (yes/no)	Categorical
region	Residential area	Categorical

Target Variable:

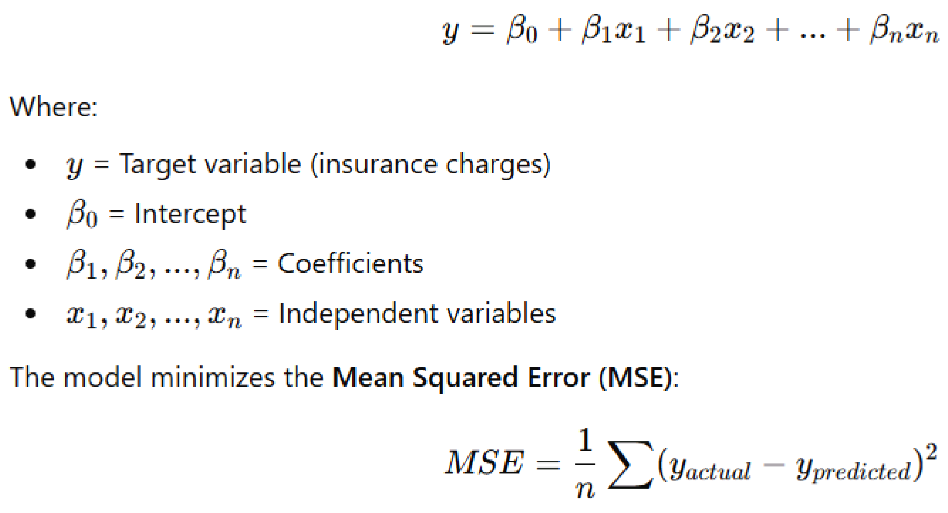
charges – Medical insurance cost (continuous numerical value)

Key Observations:

- The dataset contains no missing values.
- The feature *smoker* shows strong correlation with charges.
- Suitable for regression since target variable is continuous.

3. Mathematical Formulation of Linear Regression

Multiple Linear Regression models the relationship between dependent and independent variables as:



$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$$

Where:

- y = Target variable (insurance charges)
- β_0 = Intercept
- $\beta_1, \beta_2, \ldots, \beta_n$ = Coefficients
- $x_1, x_2, \ldots, x_n$ = Independent variables

The model minimizes the **Mean Squared Error (MSE)**:

$$MSE = \frac{1}{n} \sum (y_{actual} - y_{predicted})^2$$

4. Algorithm Limitations

Linear Regression has the following limitations:

1. Assumes linear relationship between variables.
2. Sensitive to outliers.
3. Cannot capture complex nonlinear patterns.
4. Requires independence between features (multicollinearity reduces performance).
5. Not suitable for categorical target variables.

The algorithm may not perform well if:

- Data has nonlinear relationships.
- Strong multicollinearity exists.

- Dataset contains extreme outliers.

5. Methodology / Workflow

Step-by-Step Workflow:

1. Dataset collection from Kaggle.
2. Data loading using Pandas.
3. Data preprocessing:
    - ○ Checked for missing values.
    - ○ Applied One-Hot Encoding to categorical features.
4. Feature scaling using StandardScaler.
5. Splitting dataset into:
    - ○ 80% Training Data
    - ○ 20% Testing Data
6. Model training using LinearRegression().
7. Model evaluation using:
    - ○ Mean Squared Error (MSE)
    - ○ R² Score
8. Visualization:
    - ○ Actual vs Predicted scatter plot
    - ○ Correlation heatmap

6. Performance Analysis

Results Obtained:

- **Mean Squared Error (MSE):** 33,596,915.85
- **Root Mean Squared Error (RMSE):** ≈ 5800
- **R² Score:** 0.78

Interpretation:

- The R² score of 0.78 indicates that 78% of the variation in insurance charges is explained by the model.
- RMSE of approximately 5800 means the model predicts insurance charges with an average error of $5800.
- The heatmap showed strong correlation (0.79) between smoker status and insurance charges.

This indicates good predictive performance.

7. Hyperparameter Tuning

Linear Regression has limited hyperparameters.

The following parameter was considered:

- **fit_intercept** (default = True)

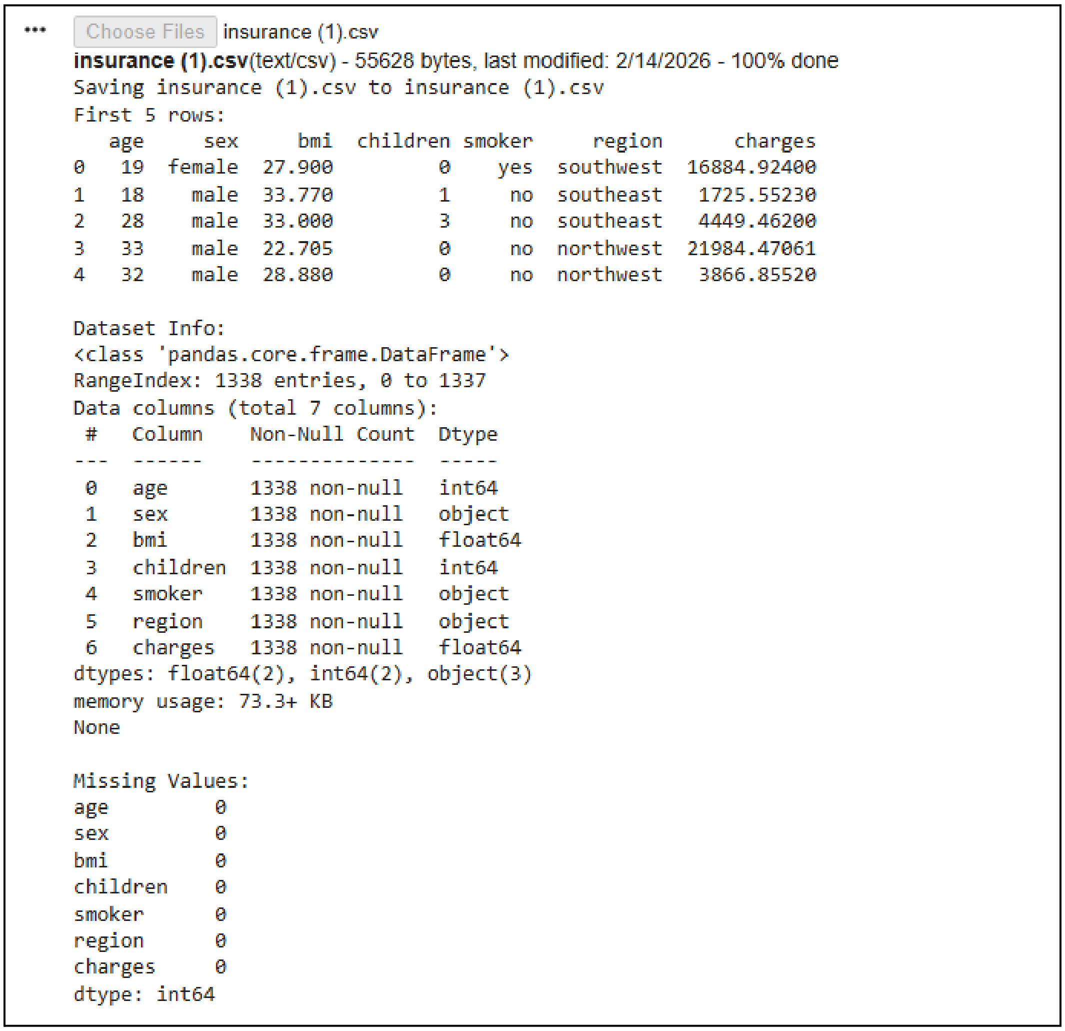
Additionally, feature scaling using StandardScaler improved numerical stability and model performance.

Since Linear Regression does not involve complex hyperparameters like tree depth or learning rate, extensive tuning was not required.

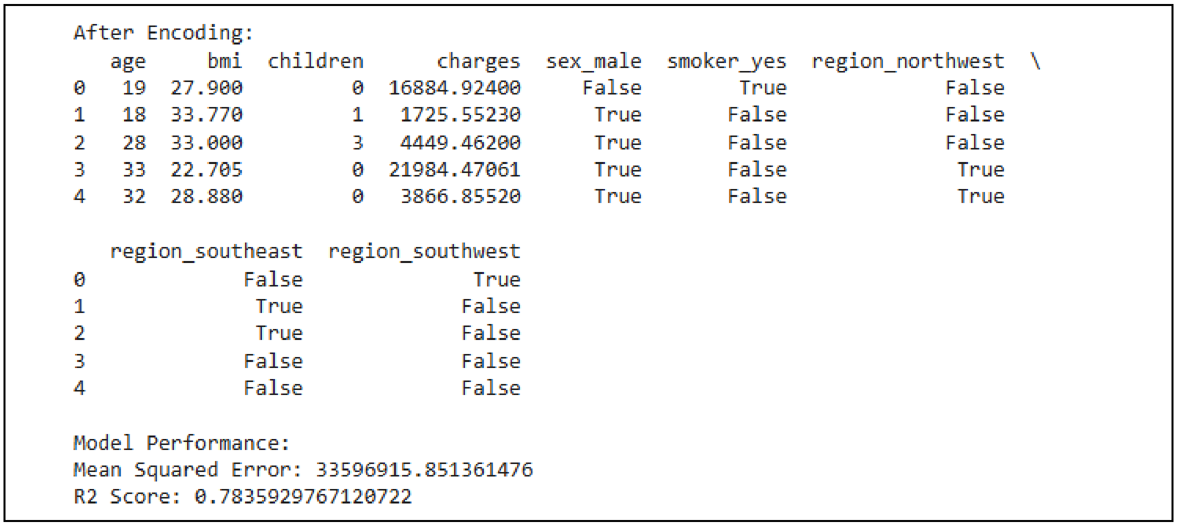
The model already achieved strong performance ($R^2 = 0.78$) without advanced tuning.

Code & Output

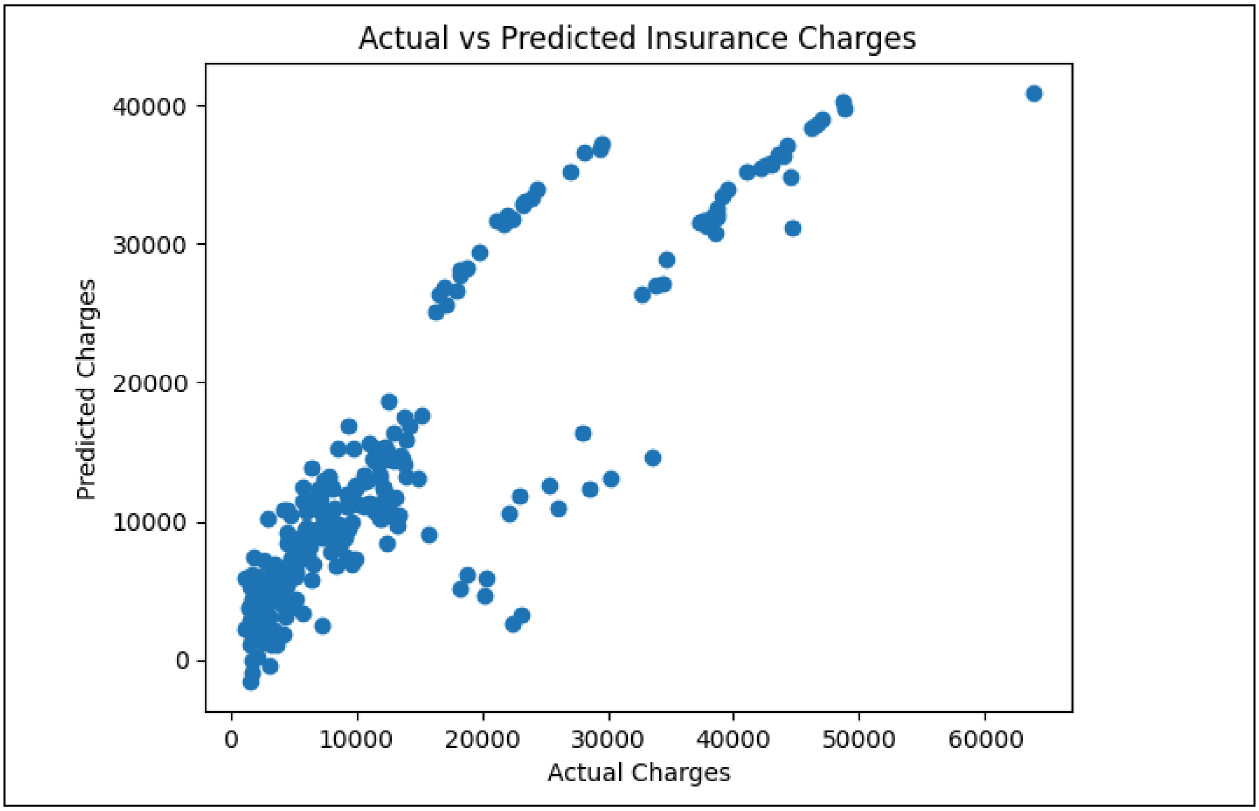
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler


from google.colab import files
uploaded = files.upload()
df = pd.read_csv("insurance.csv")
print("First 5 rows:")
print(df.head())
print("\nDataset Info:")
print(df.info())
print("\nMissing Values:")
print(df.isnull().sum())
df = pd.get_dummies(df, drop_first=True)
print("\nAfter Encoding:")
print(df.head())
X = df.drop("charges", axis=1)
y = df["charges"]
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("\nModel Performance:")
print("Mean Squared Error:", mse)
print("R2 Score:", r2)
plt.figure()
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Charges")

plt.ylabel("Predicted Charges")

plt.title("Actual vs Predicted Insurance Charges")

plt.show()

plt.figure(figsize=(10,8))

sns.heatmap(df.corr(), annot=True, cmap="coolwarm")

plt.title("Correlation Heatmap")

plt.show()
```

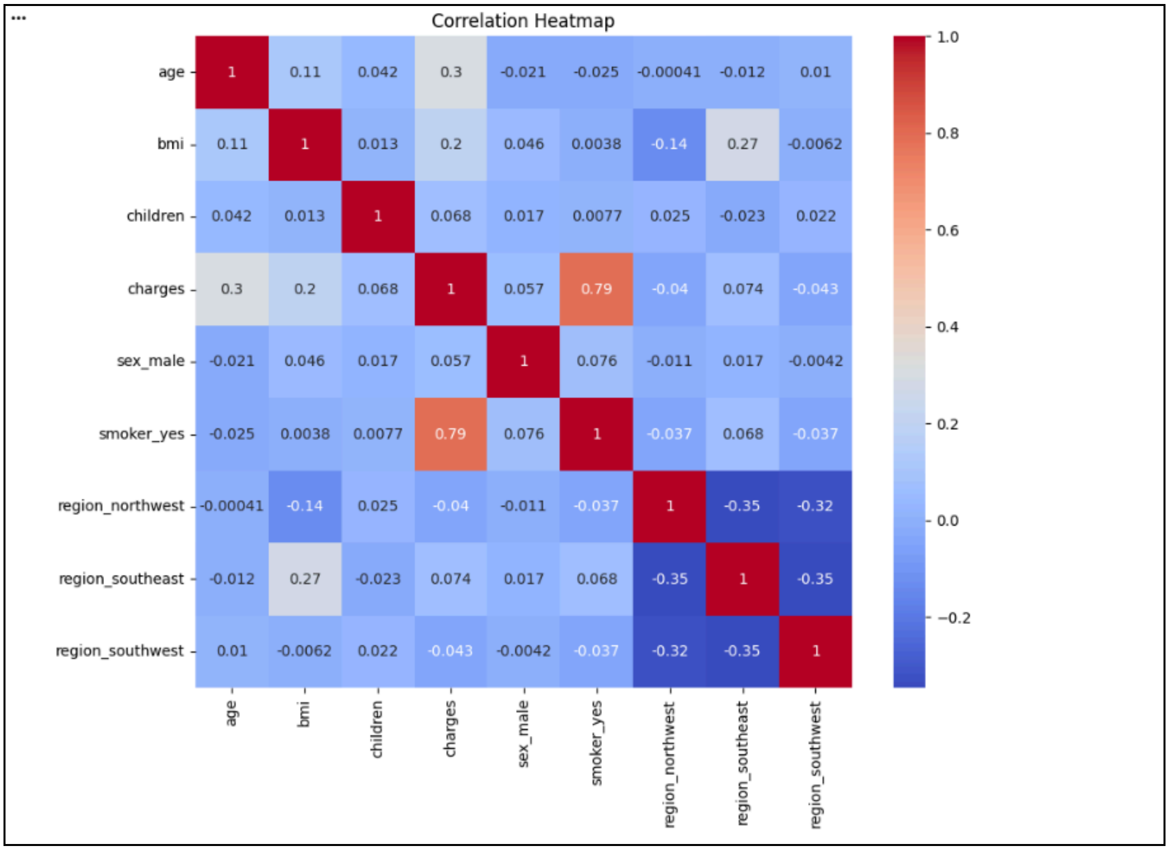


```
insurance (1).csv
insurance (1).csv(text/csv) - 55628 bytes, last modified: 2/14/2026 - 100% done
Saving insurance (1).csv to insurance (1).csv
First 5 rows:
   age     sex     bmi  children smoker     region      charges
0   19  female  27.900         0    yes  southwest  16884.92400
1   18    male  33.770         1     no  southeast   1725.55230
2   28    male  33.000         3     no  southeast   4449.46200
3   33    male  22.705         0     no  northwest  21984.47061
4   32    male  28.880         0     no  northwest   3866.85520

Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1338 non-null   int64
 1   sex       1338 non-null   object
 2   bmi       1338 non-null   float64
 3   children  1338 non-null   int64
 4   smoker    1338 non-null   object
 5   region    1338 non-null   object
 6   charges   1338 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 73.3+ KB
None

Missing Values:
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```



```
After Encoding:
   age     bmi  children      charges  sex_male  smoker_yes  region_northwest  \
0   19  27.900         0  16884.92400     False        True             False
1   18  33.770         1   1725.55230      True       False             False
2   28  33.000         3   4449.46200      True       False             False
3   33  22.705         0  21984.47061      True       False              True
4   32  28.880         0   3866.85520      True       False              True

   region_southeast  region_southwest
0             False              True
1              True             False
2              True             False
3             False             False
4             False             False

Model Performance:
Mean Squared Error: 33596915.851361476
R2 Score: 0.7835929767120722
```





Conclusion

The experiment successfully implemented Multiple Linear Regression on a real-world medical insurance dataset. After preprocessing and training, the model achieved an R² score of 0.78, demonstrating strong predictive capability. The smoker feature had the highest impact on insurance charges. The experiment validates the effectiveness of Linear Regression for continuous value prediction problems.

b. Logistic Regression

1. Dataset Source

Dataset Name: IMDB Movie Review Sentiment Dataset
Source: Kaggle
File Used: `movie.csv`

The dataset contains 40,000 movie reviews labeled as positive or negative and was used exclusively for this experiment.

2. Dataset Description

The dataset consists of movie reviews along with their sentiment labels.

Dataset Size:

- Total records: **40,000**
- Total columns: **2**

Features:

Column	Description	Type
text	Movie review text	Text
label	Sentiment (0 = Negative, 1 = Positive)	Binary

Target Variable:

label – Binary sentiment classification

- 0 → Negative Review
- 1 → Positive Review

Characteristics:

- Balanced dataset
- No missing values
- Large textual dataset suitable for NLP classification
- Requires text vectorization before model training

3. Mathematical Formulation of Logistic Regression

Logistic Regression is used for binary classification problems.

The model predicts probability using the **sigmoid function**:

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

Where:

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

The output probability lies between 0 and 1.

Decision rule:

- If probability ≥ 0.5 → Class 1 (Positive)
- If probability < 0.5 → Class 0 (Negative)

4. Algorithm Limitations

Logistic Regression has the following limitations:

1. Assumes linear relationship between features and log-odds.
2. Cannot capture complex nonlinear patterns.
3. Sensitive to irrelevant features.
4. Requires proper feature engineering for text data.
5. May underperform compared to deep learning models in advanced NLP tasks.

It may not perform well when:

- Data is highly nonlinear
- Classes are highly imbalanced
- Features are poorly engineered

5. Methodology / Workflow

Step-by-Step Workflow:

1. Dataset collection from Kaggle.
2. Data loading using Pandas.
3. Checked for missing values.
4. Split dataset into:
   - 80% Training Data
   - 20% Testing Data
5. Converted text data into numerical format using **TF-IDF Vectorizer**.
6. Trained Logistic Regression model.
7. Evaluated performance using:
   - Accuracy
   - Precision
   - Recall
   - F1-score
   - Confusion Matrix
8. Performed Hyperparameter Tuning using GridSearchCV.
9. Selected best model based on cross-validation accuracy.

6. Performance Analysis

Results Obtained:

- **Accuracy:** 88.31%
- **Precision:** 0.88 – 0.89
- **Recall:** 0.87 – 0.89
- **F1-score:** 0.88
- Balanced performance across both classes

Confusion Matrix Summary:

- Correct Negative Predictions: 3456
- Correct Positive Predictions: 3609
- Total test samples: 8000

The model correctly classified approximately 88% of movie reviews.

Interpretation:

- The model performs well for both positive and negative classes.
- Precision and recall values are balanced.
- No significant class bias is observed.
- Logistic Regression proves effective for text classification when combined with TF-IDF.

7. Hyperparameter Tuning

Hyperparameter tuning was performed using **GridSearchCV** to identify the optimal regularization parameter (C).

The following values were tested:
C = {0.01, 0.1, 1, 10, 100}

Using 5-fold cross-validation:

- Best value found: **C = 1**
- Tuned Accuracy: **88.31%**

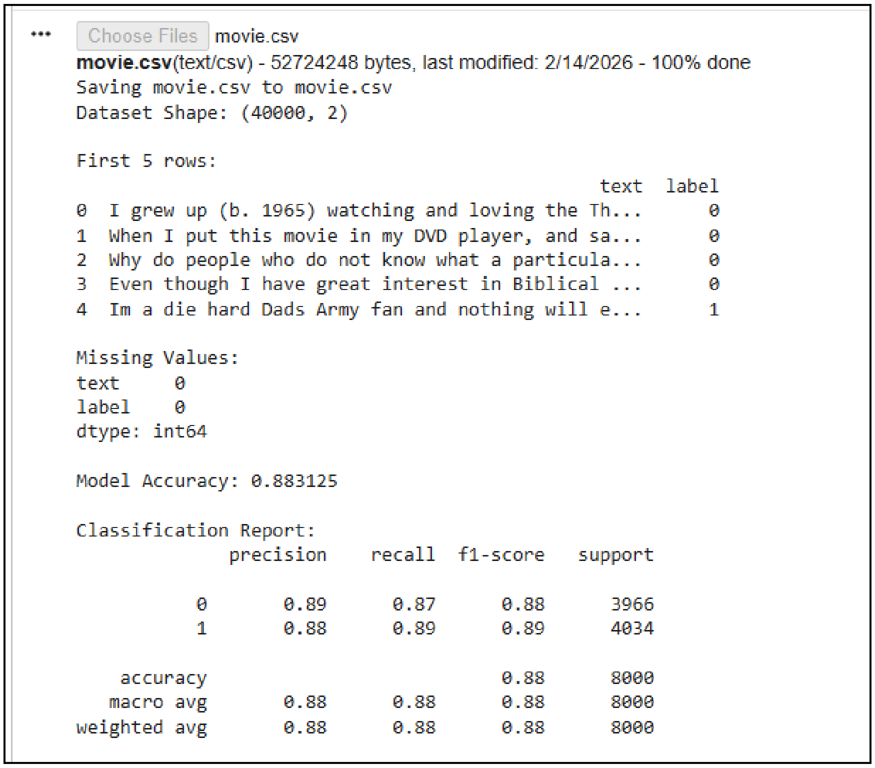
Since the tuned accuracy matched the baseline model accuracy, it confirms that the default hyperparameter setting was already optimal.

Code & Output

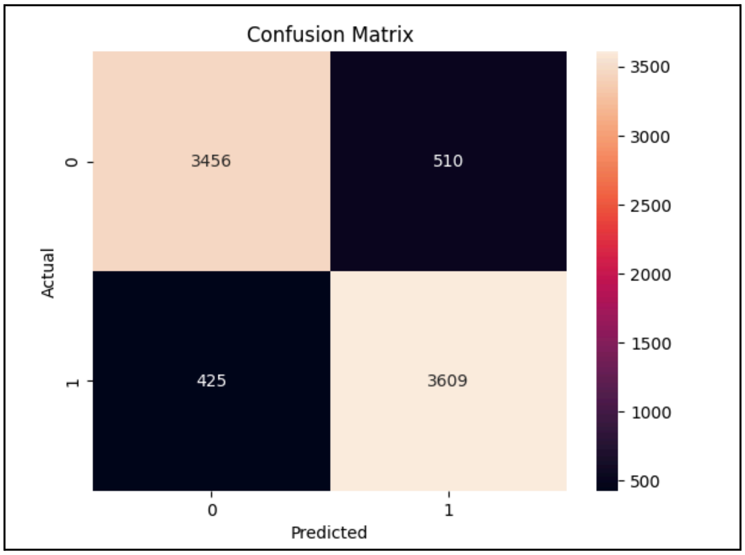
```
# Step 1: Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
# Step 2: Upload Dataset
from google.colab import files
uploaded = files.upload()
# Step 3: Load Dataset
df = pd.read_csv("movie.csv")
print("Dataset Shape:", df.shape)
print("\nFirst 5 rows:")
print(df.head())

print("\nMissing Values:")
print(df.isnull().sum())
# Step 4: Define Features and Target
X = df["text"]
y = df["label"]
```

```
# Step 5: Train-Test Split (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
# Step 6: Convert Text to Numerical Features using TF-IDF
vectorizer = TfidfVectorizer(
    max_features=5000,
    stop_words='english'
)
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)
# Step 7: Train Logistic Regression Model
model = LogisticRegression(max_iter=1000)
model.fit(X_train_tfidf, y_train)
# Step 8: Make Predictions
y_pred = model.predict(X_test_tfidf)
# Step 9: Evaluate Model
accuracy = accuracy_score(y_test, y_pred)
print("\nModel Accuracy:", accuracy)
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
# Step 10: Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure()
```

```
sns.heatmap(cm, annot=True, fmt='d')

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.title("Confusion Matrix")

plt.show()
```



```
movie.csv(text/csv) - 52724248 bytes, last modified: 2/14/2026 - 100% done
Saving movie.csv to movie.csv
Dataset Shape: (40000, 2)

First 5 rows:
                                                text  label
0  I grew up (b. 1965) watching and loving the Th...      0
1  When I put this movie in my DVD player, and sa...      0
2  Why do people who do not know what a particula...      0
3  Even though I have great interest in Biblical ...      0
4  Im a die hard Dads Army fan and nothing will e...      1

Missing Values:
text     0
label    0
dtype: int64

Model Accuracy: 0.883125

Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.87      0.88      3966
           1       0.88      0.89      0.89      4034

    accuracy                           0.88      8000
   macro avg       0.88      0.88      0.88      8000
weighted avg       0.88      0.88      0.88      8000
```



Conclusion

The experiment successfully implemented Logistic Regression for sentiment classification on a large movie review dataset. Text data was converted into numerical form using TF-IDF vectorization. The model achieved an accuracy of 88.31%, demonstrating strong predictive performance. Hyperparameter tuning confirmed the robustness of the default configuration. Logistic Regression proved to be an effective and efficient algorithm for binary text classification tasks.