Monica Mewani D15B/42

AIM

Implementation of Linear and Logistic Regression on Real-World Datasets

THEORY

📌 Regression

Regression is a supervised machine learning technique used to model the relationship between independent variables (features) and a dependent variable (target). It helps in predicting unknown values based on known data.

There are mainly two types of regression used in this experiment:

🔹 Linear Regression

Linear Regression is used when the output variable is **continuous**.

It assumes a linear relationship between input variables and output.

Mathematical Model:

y=mx+cy = mx + cy=mx+c

For multiple variables:

y=β0+β1x1+β2x2+...+βnxny = β_0 + β_1x_1 + β_2x_2 + ... + β_nx_ny=β0+β1x1+β2x2+...+βnxn

Where:

- y → predicted value

- x → input features
- β → coefficients
- β_0 → intercept

In this experiment:

Target variable: **BMI**
Type: Continuous

Hence, Linear Regression is suitable.

Working of Linear Regression:

1. Initializes weights.
2. Fits best straight line minimizing error.
3. Uses Mean Squared Error to optimize parameters.
4. Predicts BMI for unseen data.

Evaluation Metrics:

- MAE (Mean Absolute Error)
- MSE (Mean Squared Error)
- RMSE (Root Mean Squared Error)
- R^2 Score (Goodness of fit)

Lower MAE/MSE and higher R^2 indicate better model performance.

🔹 Logistic Regression

Monica Mewani D15B/42

Logistic Regression is used for **binary classification problems**.

Instead of predicting continuous values, it predicts probabilities between 0 and 1 using the **Sigmoid function**.

Sigmoid Function:

σ(x)=11+e−xσ(x) = \frac{1}{1 + e^{-x}}σ(x)=1+e−x1

Output > 0.5 → Class 1
Output < 0.5 → Class 0

In this experiment:

Target variable: **Stroke**

0 → No Stroke
1 → Stroke

Working of Logistic Regression:

1. Applies linear equation.
2. Passes result through sigmoid.
3. Converts probability into binary output.
4. Uses cross-entropy loss for optimization.

Evaluation Metrics:

- Accuracy
- Confusion Matrix
- Precision
- Recall
- F1-Score

These metrics help analyze classification performance.

📊 Dataset Description

The healthcare stroke dataset contains patient health records including:

- Age
- Gender
- Hypertension
- Heart disease
- BMI
- Smoking status

Linear Regression predicts BMI, while Logistic Regression predicts stroke occurrence.

Categorical values are encoded and missing BMI values are handled using mean imputation.

⚠ Limitations

Linear Regression:

- Assumes linearity
- Sensitive to outliers
- Cannot model complex patterns

Logistic Regression:

- Assumes linear decision boundary

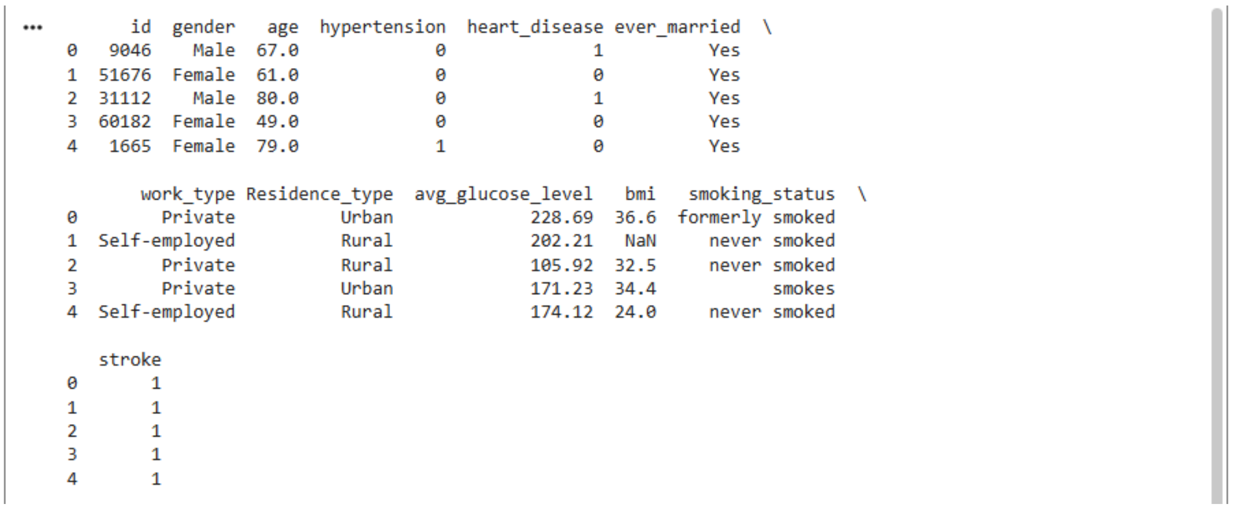
- Struggles with imbalanced datasets
- Limited for non-linear relationship

CODE

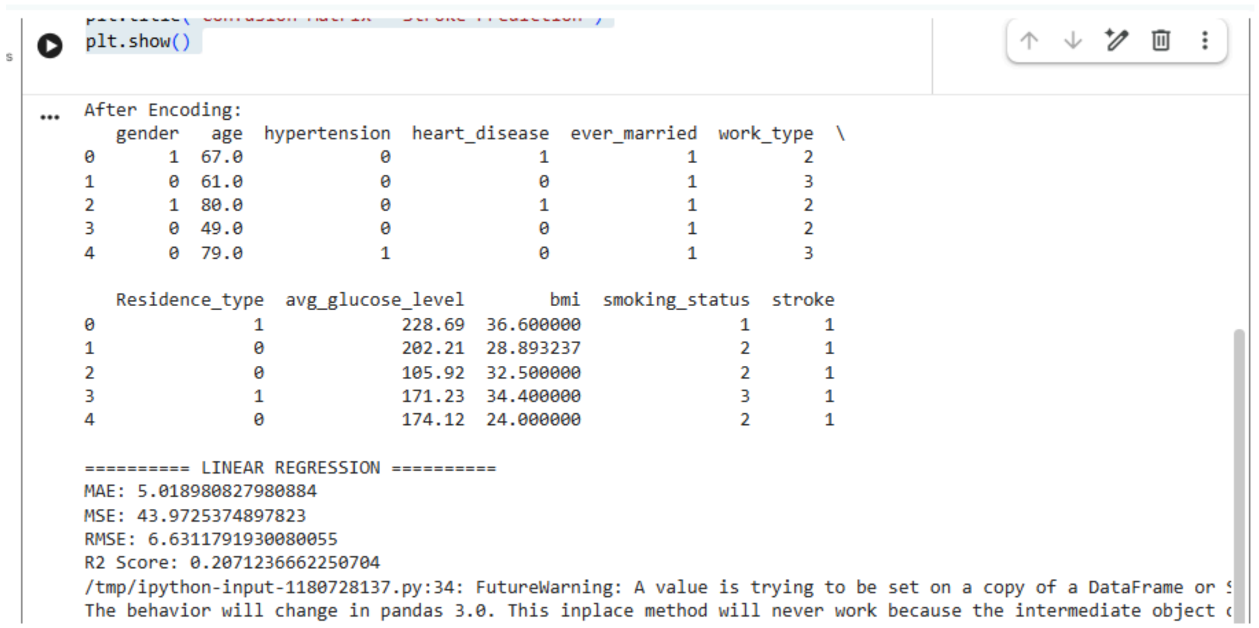
```
# ======================================
# STEP 1: Import Libraries
# ======================================

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.linear_model import LinearRegression, LogisticRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
from sklearn.preprocessing import LabelEncoder


# ======================================
# STEP 2: Load Dataset
# ======================================

df = pd.read_csv("healthcare-dataset-stroke-data.csv")

print(df.head())


# ======================================
# STEP 3: Data Cleaning
# ======================================
# Drop ID column
df.drop("id", axis=1, inplace=True)

# Fill missing BMI with mean
df["bmi"].fillna(df["bmi"].mean(), inplace=True)


# =====================================
# STEP 4: Encode Categorical Columns
# =====================================

le = LabelEncoder()

cat_cols =
["gender","ever_married","work_type","Residence_type","smoking_status"]

for col in cat_cols:
    df[col] = le.fit_transform(df[col])


print("\nAfter Encoding:")
print(df.head())


# =========================================================
# PART A — LINEAR REGRESSION (Predict BMI)
# =========================================================

print("\n========== LINEAR REGRESSION ==========")

X = df.drop("bmi", axis=1)
y = df["bmi"]

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

lr = LinearRegression()
lr.fit(X_train, y_train)
```

```
y_pred_lr = lr.predict(X_test)

print("MAE:", mean_absolute_error(y_test, y_pred_lr))
print("MSE:", mean_squared_error(y_test, y_pred_lr))
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred_lr)))
print("R2 Score:", r2_score(y_test, y_pred_lr))

plt.scatter(y_test, y_pred_lr)
plt.xlabel("Actual BMI")
plt.ylabel("Predicted BMI")
plt.title("Linear Regression: BMI Prediction")
plt.show()


# =========================================================
# PART B — LOGISTIC REGRESSION (Predict Stroke)
# =========================================================

print("\n========== LOGISTIC REGRESSION ==========")

X = df.drop("stroke", axis=1)
y = df["stroke"]

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

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

y_pred_log = log_model.predict(X_test)

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

sns.heatmap(confusion_matrix(y_test, y_pred_log), annot=True, fmt="d")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix - Stroke Prediction")
plt.show()
```

OUTPUT

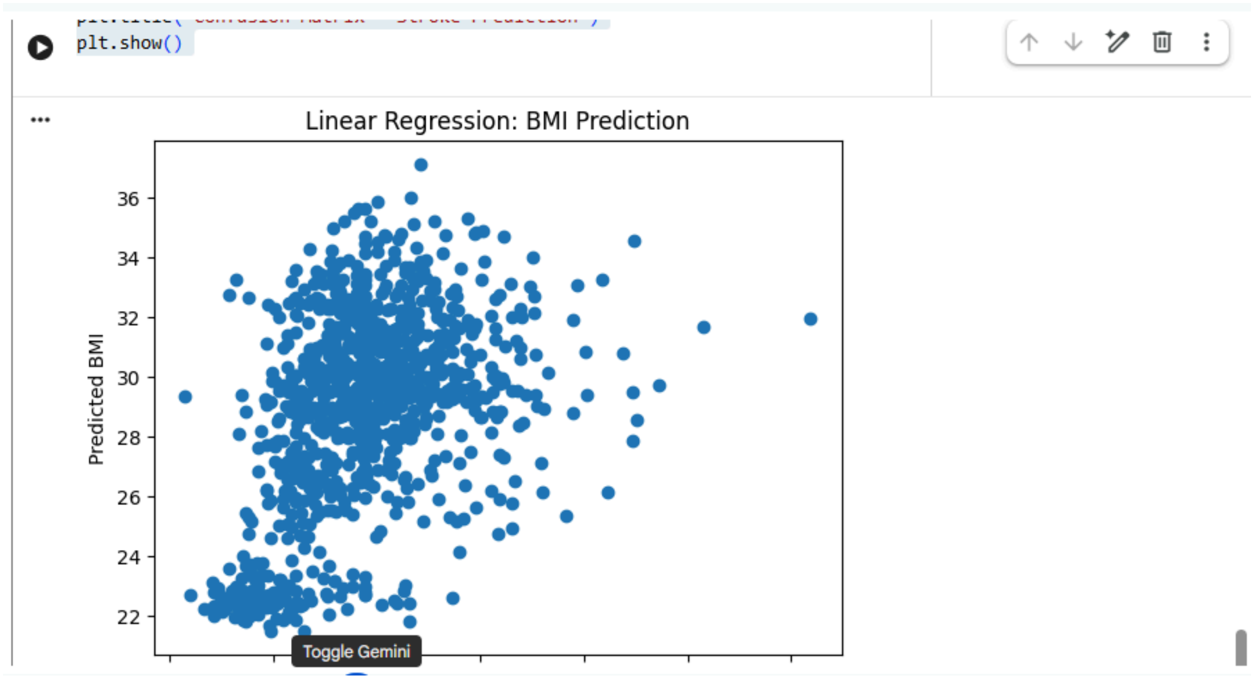


```
      id  gender   age  hypertension  heart_disease ever_married  \
0   9046    Male  67.0             0              1          Yes
1  51676  Female  61.0             0              0          Yes
2  31112    Male  80.0             0              1          Yes
3  60182  Female  49.0             0              0          Yes
4   1665  Female  79.0             1              0          Yes

       work_type Residence_type  avg_glucose_level   bmi   smoking_status  \
0        Private          Urban             228.69  36.6  formerly smoked
1  Self-employed          Rural             202.21   NaN     never smoked
2        Private          Rural             105.92  32.5     never smoked
3        Private          Urban             171.23  34.4           smokes
4  Self-employed          Rural             174.12  24.0     never smoked

   stroke
0       1
1       1
2       1
3       1
4       1
```

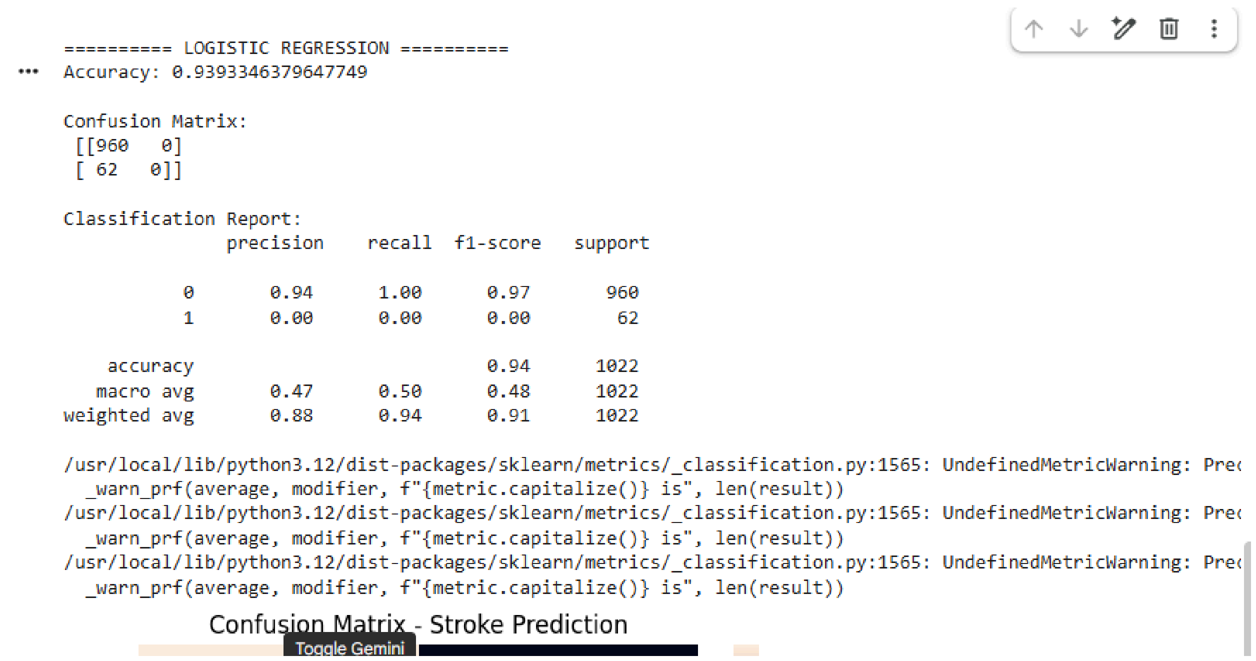


```
plt.show()
```

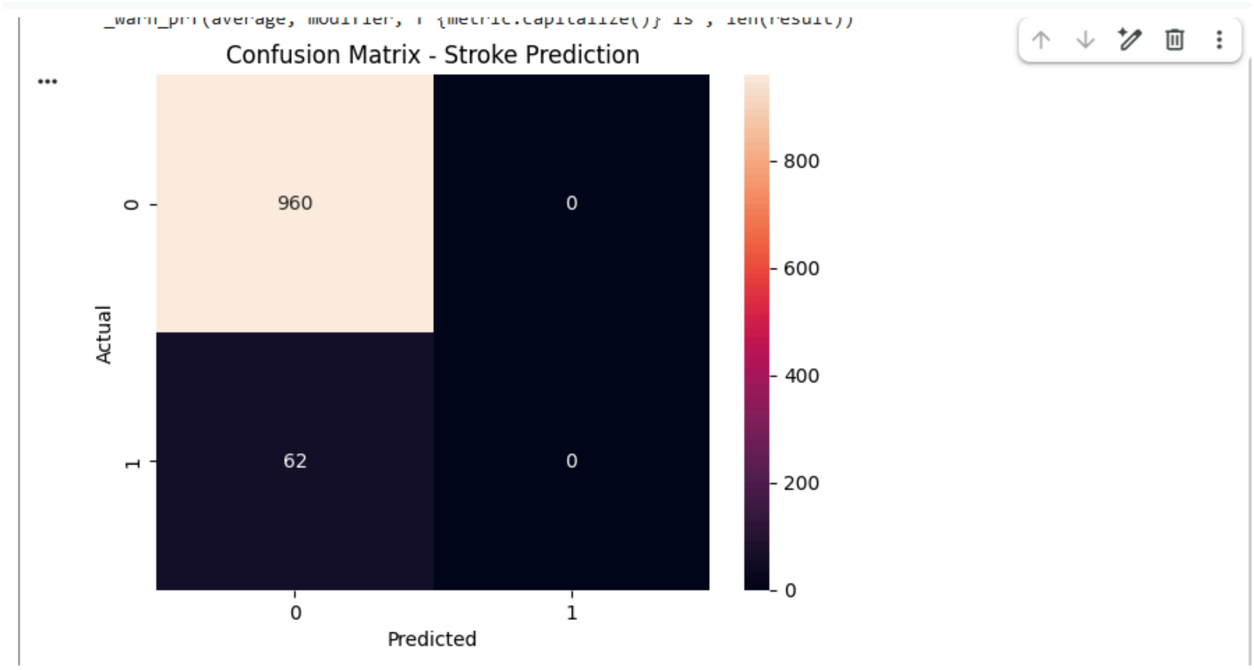
```
After Encoding:
   gender   age  hypertension  heart_disease  ever_married  work_type  \
0       1  67.0             0              1             1          2
1       0  61.0             0              0             1          3
2       1  80.0             0              1             1          2
3       0  49.0             0              0             1          2
4       0  79.0             1              0             1          3

   Residence_type  avg_glucose_level        bmi  smoking_status  stroke
0               1             228.69  36.600000               1       1
1               0             202.21  28.893237               2       1
2               0             105.92  32.500000               2       1
3               1             171.23  34.400000               3       1
4               0             174.12  24.000000               2       1

========== LINEAR REGRESSION ==========
MAE: 5.018980827980884
MSE: 43.9725374897823
RMSE: 6.6311791930080055
R2 Score: 0.2071236662250704
/tmp/ipython-input-1180728137.py:34: FutureWarning: A value is trying to be set on a copy of a DataFrame or S
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object c
```



```
plt.show()
```



```
========== LOGISTIC REGRESSION ==========
Accuracy: 0.9393346379647749

Confusion Matrix:
 [[960   0]
 [ 62   0]]

Classification Report:
               precision    recall  f1-score   support

           0       0.94      1.00      0.97       960
           1       0.00      0.00      0.00        62

    accuracy                           0.94      1022
   macro avg       0.47      0.50      0.48      1022
weighted avg       0.88      0.94      0.91      1022

/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Pre
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Pre
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Pre
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```



```
_warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```