Name : Sahil Jakhariya
Div : D15C
Batch : C
Roll No : 63

MLDL Experiment 01

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

Linear Regression:

1. Dataset Source:

- **Dataset Name: Medical Cost Personal Datasets**
- **Source:** https://www.kaggle.com/datasets/iammustafatz/diabetes-prediction-dataset
- **Repository Owner: Miri Choi**

This dataset contains medical insurance records used for practicing regression and machine learning workflows. The data represents realistic relationships between personal attributes and medical insurance charges, making it suitable for exploratory data analysis (EDA), feature engineering, and regression modeling.

● Size: 1338 samples (rows) × multiple attributes (columns)

● Target Variable:

○ charges (Continuous) – Individual medical insurance cost billed by the insurance company.

● **Key Features:**

1. age – Age of the policyholder.
2. sex – Gender of the individual (male or female).
3. bmi – Body Mass Index, indicating body fat based on height and weight.
4. children – Number of dependents covered under insurance.
5. smoker – Indicates whether the individual is a smoker or non-smoker.
6. region – Residential region of the individual (southeast, southwest, northeast, northwest).

The dataset is clean, beginner-friendly, and contains both numerical and categorical features. Minimal preprocessing such as encoding categorical variables is required before applying regression algorithms.

3. Mathematical Formulation of the Algorithm

Multiple Linear Regression models the relationship between several independent variables and a continuous dependent variable. It predicts the medical insurance charges based on input features such as age, BMI, smoking status, and other personal attributes by fitting a linear equation to the observed data.

4. Algorithm Limitations

1. Linearity Assumption: The model assumes a linear relationship between input features and medical insurance charges.
2. Sensitive to Outliers: Extremely high or low medical charges may affect prediction accuracy.
3. Multicollinearity: Highly correlated features such as age and BMI may reduce model stability and affect coefficient interpretation.

5. Methodology / Workflow

1. Load CSV dataset.
2. Select input features and target variable (charges).
3. Convert categorical variables into numerical form using encoding.
4. Split dataset into training and testing sets (80:20).
5. Train Multiple Linear Regression model.
6. Predict medical insurance charges.
7. Evaluate model performance using MSE and R^2 score.

6. Performance Analysis

● Metric 1: Mean Squared Error (MSE)

○ Measures the average squared difference between actual and predicted insurance charges.

● Metric 2: R-Squared (R^2)

○ Indicates the percentage of variance in medical insurance charges explained by the input features.

Higher R^2 and lower MSE indicate better model performance.

7. Hyperparameter Tuning

Standard Linear Regression has no tunable hyperparameters.

To improve performance, Ridge Regression (L2 Regularization) can be applied.

- Tuned Parameter: alpha
- Tested Values: [0.1, 1, 10]

- Helps reduce overfitting and improve model generalization.

Logistic Regression:
2. Dataset Source:

- **Dataset Name: Diabetes prediction dataset**
- **Source:** https://www.kaggle.com/datasets/iammustafatz/diabetes-prediction-dataset
- **Repository Owner: Mohammed Mustafa**

3. Dataset Description:

This dataset contains **2,500 weather observations** and is designed for beginners to practice **classification using machine learning**.
The dataset is used to predict whether **rainfall will occur** based on several atmospheric and environmental conditions.

- **Size:** 2,500 samples (rows) × multiple attributes (columns)
- **Target Variable:**
    - Rain (Binary)

- 1 → Rainfall occurs
- 0 → No rainfall

- **Key Features:**

1. Temperature – Current temperature
2. Humidity – Moisture content in air
3. WindSpeed – Speed of wind
4. Pressure – Atmospheric pressure
5. CloudCover – Percentage of cloud cover
6. Visibility – Distance that can be clearly seen
7. DewPoint – Temperature at which condensation occurs

The dataset is **clean, simple, and beginner-friendly**, making it suitable for experimenting with classification algorithms such as Logistic Regression, Decision Trees, and Random Forests.

4. Mathematical Formulation of the Algorithm:

Logistic Regression estimates the probability that an instance belongs to a particular class (Rain or No Rain).

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

Where:

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

Decision Rule:

$$\hat{y} = \begin{cases} 1, & \text{if } \sigma(z) \geq 0.5 \\ 0, & \text{if } \sigma(z) < 0.5 \end{cases}$$

5. Algorithm Limitations:

1. **Linear Decision Boundary:** Cannot model complex non-linear relationships.
2. **Sensitive to Feature Scaling:** Features should be standardized.
3. **Performance Drops with Highly Correlated Features.**

6. Methodology / Workflow:

1. Load weather dataset
2. Separate features and target (Rain)
3. Apply feature scaling (StandardScaler)
4. Split dataset into training and testing sets (80:20)
5. Train Logistic Regression model
6. Predict rainfall
7. Evaluate performance

7. Performance Analysis:

- **Metric 1: Accuracy**
    - Measures overall correctness of predictions
- **Metric 2: Confusion Matrix**
    - Shows True Positives, False Positives, True Negatives, False Negatives

Higher accuracy and balanced confusion matrix indicate better model performance.

8. Hyperparameter Tuning:

- Parameter Tuned: `C` (Inverse of Regularization Strength)
- Tested Values: [0.01, 0.1, 1, 10, 100]

Impact:

- Low C → Strong regularization (simpler model)
- High C → Weak regularization (complex model)

Code And Output:

Linear Regression

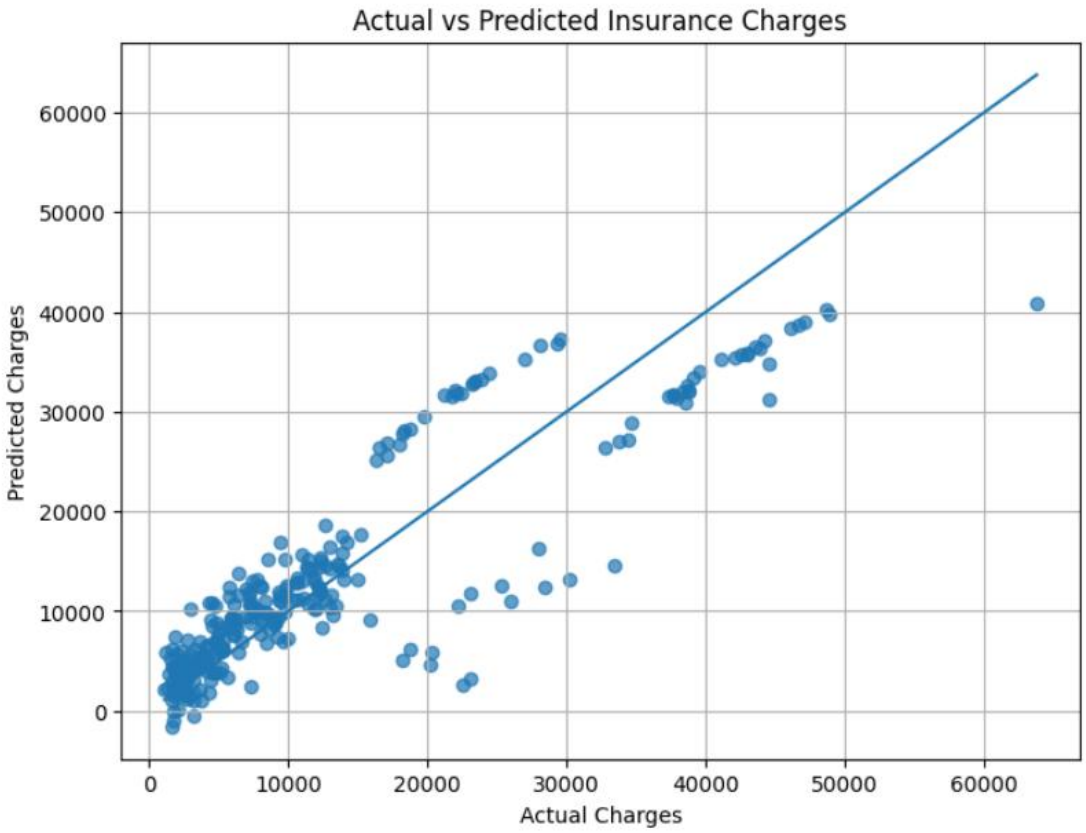
```
# Import libraries
from google.colab import files
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics

# Upload dataset
uploaded = files.upload()

# Load dataset
df = pd.read_csv(list(uploaded.keys())[0])

print("===== Dataset Preview =====")
print(df.head())

# Convert categorical columns to numeric
df = pd.get_dummies(df, drop_first=True)

print("\n===== Processed Dataset =====")
print(df.head())

# Features and target
X = df.drop("charges", axis=1)
y = df["charges"]
```

```
print("\n===== Features Used =====")
print(X.columns)

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

# Train model
model = LinearRegression()
model.fit(X_train, y_train)

# Prediction
y_pred = model.predict(X_test)

# Model evaluation
mae = metrics.mean_absolute_error(y_test, y_pred)
mse = metrics.mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = metrics.r2_score(y_test, y_pred)

print("\n===== Model Performance =====")
print(f"Mean Absolute Error (MAE): {mae:.4f}")
print(f"Mean Squared Error (MSE): {mse:.4f}")
print(f"Root Mean Squared Error (RMSE): {rmse:.4f}")
print(f"R-squared Score (R2): {r2:.4f} ({r2*100:.2f}%)")

# Regression equation details
print("\n===== Regression Equation Details =====")
```

```
print(f"Intercept (b0): {model.intercept_:.4f}")


coeff_df = pd.DataFrame(model.coef_, X.columns,
columns=["Coefficient"])
print(coeff_df)


# Actual vs Predicted Plot
plt.figure(figsize=(8,6))
plt.scatter(y_test, y_pred, alpha=0.7)
plt.plot([y.min(), y.max()], [y.min(), y.max()])
plt.xlabel("Actual Charges")
plt.ylabel("Predicted Charges")
plt.title("Actual vs Predicted Insurance Charges")
plt.grid(True)
plt.show()
```

```
===== Model Performance =====
Mean Absolute Error (MAE): 4181.1945
Mean Squared Error (MSE): 33596915.8514
Root Mean Squared Error (RMSE): 5796.2847
R-squared Score (R2): 0.7836 (78.36%)

===== Regression Equation Details =====
Intercept (b0): -11931.2191
                  Coefficient
age                 256.975706
bmi                 337.092552
children            425.278784
sex_male            -18.591692
smoker_yes        23651.128856
region_northwest   -370.677326
region_southeast   -657.864297
region_southwest   -809.799354
```

...



Logistic Regression (Diabetes Dataset)

```
# Import libraries

from google.colab import files

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 StandardScaler

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, recall_score, f1_score



# Upload dataset

uploaded = files.upload()



# Load dataset

df = pd.read_csv(list(uploaded.keys())[0])



print("===== Dataset Preview =====")

print(df.head())



# Convert categorical values if present

for col in df.columns:

    if df[col].dtype == "object":

        df[col] = df[col].astype("category").cat.codes



# Assume last column is target (same logic as your experiment file)

target = df.columns[-1]
```

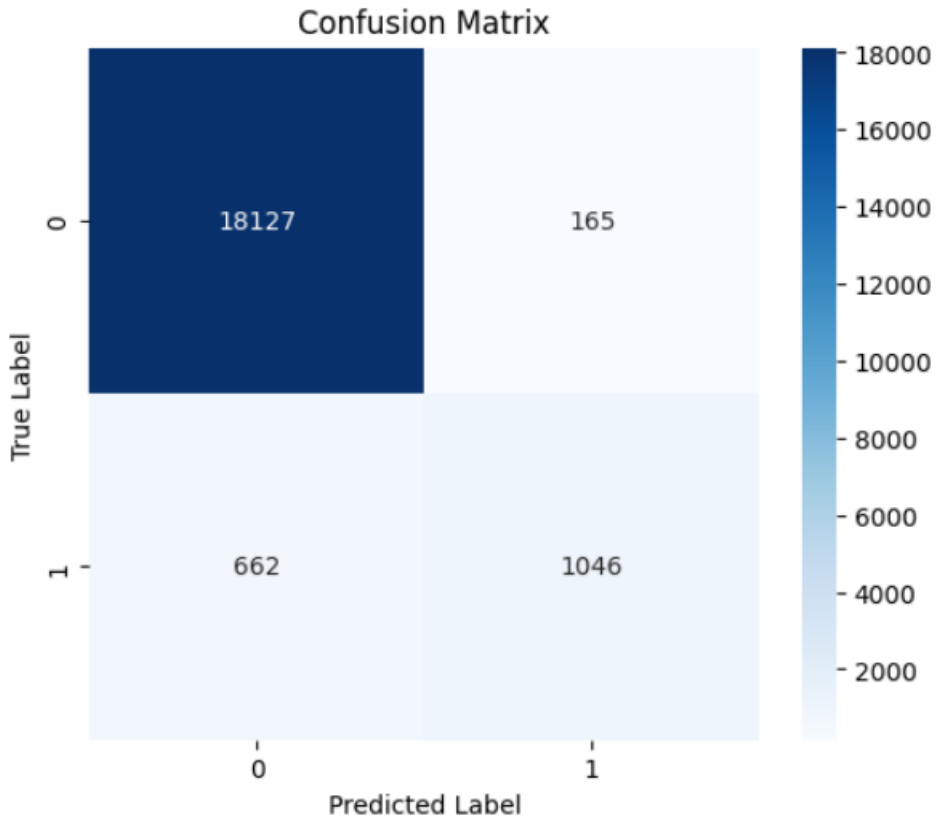
```
# Features and target

X = df.drop(target, axis=1)

y = df[target]


# Keep only numeric features

X = X.select_dtypes(include=[np.number])


print("\n===== Features Used =====")

print(X.columns)

print("\nTarget Variable:", target)


# 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 (important for logistic regression)

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)


# Train model
```

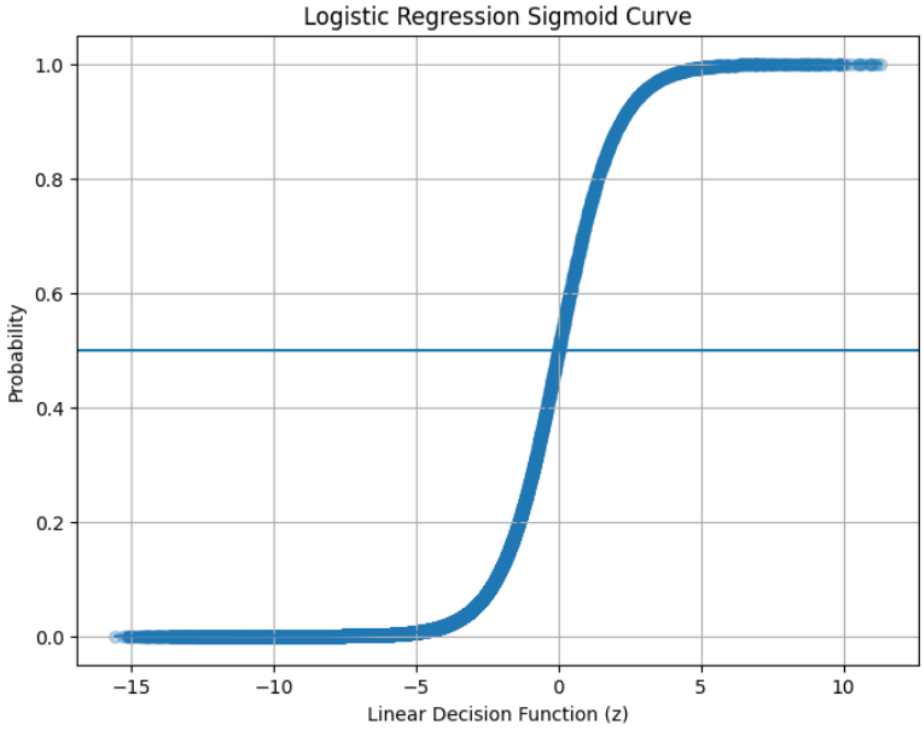
```
model = LogisticRegression(max_iter=10000)

model.fit(X_train, y_train)


# Prediction

y_pred = model.predict(X_test)


# Performance metrics

acc = accuracy_score(y_test, y_pred)

prec = precision_score(y_test, y_pred)

rec = recall_score(y_test, y_pred)

f1 = f1_score(y_test, y_pred)


print("\n===== Model Performance =====")

print(f"Accuracy: {acc:.4f} ({acc*100:.2f}%)")

print(f"Precision: {prec:.4f} ({prec*100:.2f}%)")

print(f"Recall: {rec:.4f} ({rec*100:.2f}%)")

print(f"F1 Score: {f1:.4f} ({f1*100:.2f}%)")


# Confusion matrix

cm = confusion_matrix(y_test, y_pred)

tn, fp, fn, tp = cm.ravel()
```

```
print("\n===== Confusion Matrix Values =====")

print(f"TP: {tp}, TN: {tn}, FP: {fp}, FN: {fn}")


# Confusion matrix plot

plt.figure(figsize=(6,5))

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

plt.xlabel("Predicted Label")

plt.ylabel("True Label")

plt.title("Confusion Matrix")

plt.show()


# Sigmoid curve visualization

z = model.decision_function(X_test)

prob = model.predict_proba(X_test)[:,1]


plt.figure(figsize=(8,6))

idx = np.argsort(z)

plt.scatter(z, prob, alpha=0.3)

plt.plot(z[idx], prob[idx])

plt.axhline(0.5)

plt.xlabel("Linear Decision Function (z)")

plt.ylabel("Probability")
```

```
plt.title("Logistic Regression Sigmoid Curve")

plt.grid(True)

plt.show()
```

```
Target Variable: diabetes

===== Model Performance =====
Accuracy: 0.9587 (95.86%)
Precision: 0.8637 (86.37%)
Recall: 0.6124 (61.24%)
F1 Score: 0.7167 (71.67%)

===== Confusion Matrix Values =====
TP: 1046, TN: 18127, FP: 165, FN: 662
```


Confusion Matrix
18127
165
662
1046
True Label
Predicted Label
0
1
18000
16000
14000
12000
10000
8000
6000
4000
2000


Logistic Regression Sigmoid Curve
Probability
Linear Decision Function (z)
1.0
0.8
0.6
0.4
0.2
0.0
−15
−10
−5
0
5
10

Conclusion:

This experiment demonstrated the practical implementation of Linear Regression for medical insurance charge prediction and Logistic Regression for diabetes prediction using Kaggle datasets. Both models successfully captured relationships between input features and their respective target variables, showing the effectiveness of supervised learning techniques for solving real-world problems.

The experiment highlighted the importance of data preprocessing, feature selection, feature scaling, and train–test splitting in building reliable machine learning models. Performance evaluation using metrics such as Mean Squared Error, R^2 score, Accuracy, and Confusion Matrix provided insights into model effectiveness. Overall, this experiment strengthened understanding of regression and classification algorithms and their application in predictive analytics in healthcare and insurance domains.