Name : Vaibhav Gupta
Div : D15C
Batch : C
Roll No : 61

MLDL Experiment 01

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

Linear Regression:

1. Dataset Source:

- **Dataset Name**: House Price Prediction Dataset
- **Source:** https://www.kaggle.com/datasets/suvidyasonawane/house-price-prediction-dataset
- **Repository Owner:** Suvidya Sonawane

2. Dataset Description:

This dataset contains 300 synthetic housing records created for practicing regression and machine learning workflows.
The data simulates realistic relationships between housing features and sale prices, making it ideal for exploratory data analysis (EDA), feature engineering, and regression modeling.

- Size: 300 samples (rows) × multiple numerical attributes (columns)

- Target Variable:
    - SalePrice (Continuous) – Final selling price of the house
- Key Features:

1. OverallQual – Overall material and finish quality of the house
2. GrLivArea – Above ground living area (square feet)
3. GarageCars – Number of cars that can fit in the garage
4. TotalBsmtSF – Total basement area (square feet)
5. YearBuilt – Year the house was constructed
6. TotRmsAbvGrd – Total rooms above ground
7. LotArea – Size of the lot in square feet

The dataset is clean, beginner-friendly, and requires no preprocessing, allowing direct application of regression algorithms.

3. Mathematical Formulation of the Algorithm:

Multiple Linear Regression models the relationship between several independent variables and a continuous dependent variable.

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

Where:

- $\hat{y}$ = Predicted house price
- β_0 = Intercept
- $\beta_1 ... \beta_n$ = Coefficients
- $x_1 ... x_n$ = Input features

Cost Function (Mean Squared Error):

$$J(\beta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

4. Algorithm Limitations:

1. **Linearity Assumption:** Assumes linear relationship between features and price.

2. **Sensitive to Outliers:** Extremely expensive houses may distort predictions.

3. **Multicollinearity:** Highly correlated features can reduce model stability.

5. Methodology / Workflow:

1. Load CSV dataset

2. Select input features and target (`SalePrice`)
3. Split dataset into training and testing sets (80:20)
4. Train Multiple Linear Regression model
5. Predict house prices
6. Evaluate using MSE and R^2 score

6. Performance Analysis:

- **Metric 1:** Mean Squared Error (MSE)
  - Measures average squared prediction error
- **Metric 2:** R-Squared (R^2)
  - Indicates percentage of variance in house prices explained by 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 generalization.

Logistic Regression:

1. Dataset Source:

- **Dataset Name:** Weather Forecast Dataset
- **Source:** https://www.kaggle.com/datasets/zeeshier/weather-forecast-dataset
- **Repository Owner:** Zeeshier

2. 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.

3. 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:

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

Decision Rule:

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

4. 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.**

5. 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

6. 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.

7. 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)

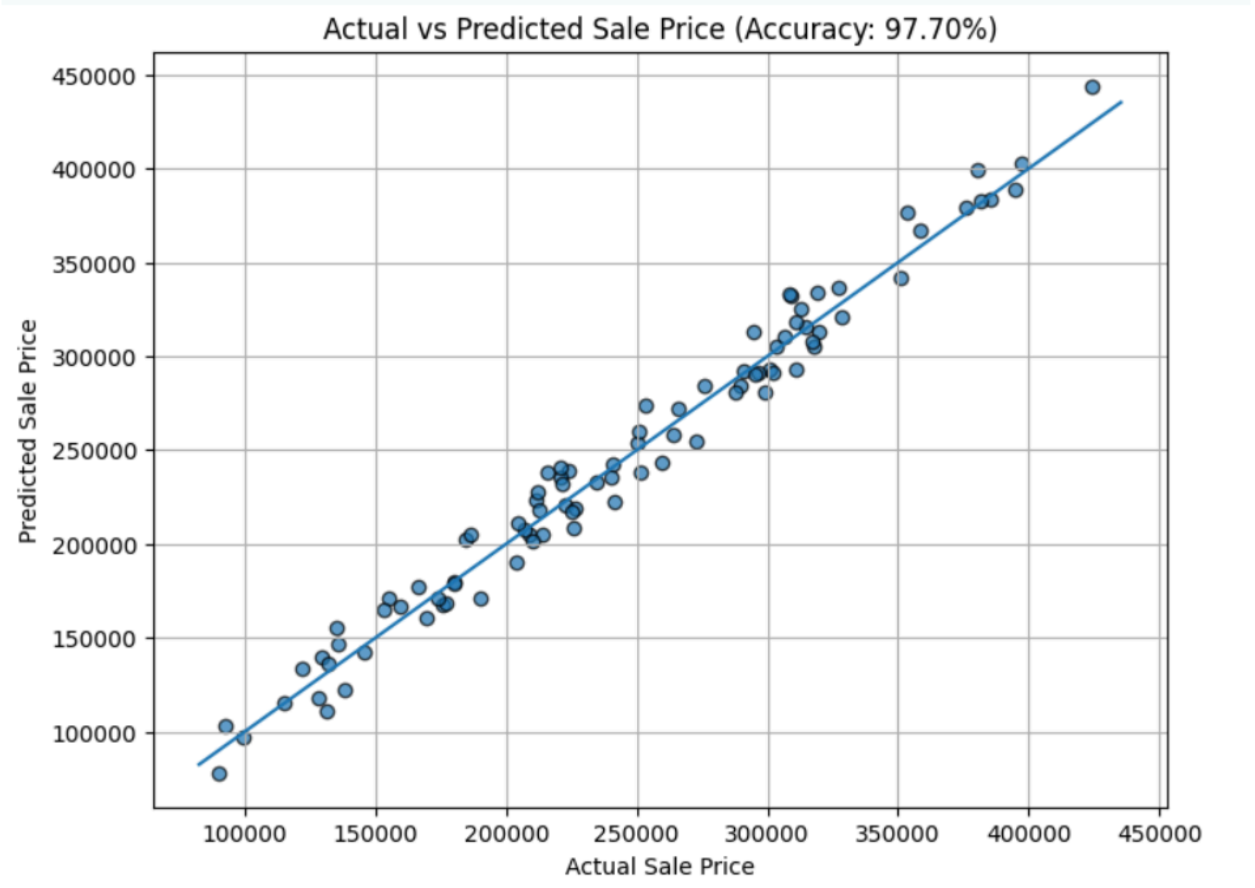
Code And Output:

```
from google.colab import files

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn import metrics
```

```
uploaded = files.upload()


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


X = df.drop('SalePrice', axis=1)

y = df['SalePrice']


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


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)


model = LinearRegression()

model.fit(X_train, y_train)


y_pred = model.predict(X_test)

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)

mape = np.mean(np.abs((y_test - y_pred) / y_test))


print("---- Model Performance Metrics ----")

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}%)")

print(f"Mean Absolute Percentage Error (MAPE): {mape:.4f} ({mape*100:.2f}%)")
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)



equation = f"SalePrice = {model.intercept_:.4f}"

for i, col in enumerate(X.columns):

    equation += f" + ({model.coef_[i]:.4f} * {col})"
```

```
print("\nMathematical Equation:")

print(equation)



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

plt.scatter(y_test, y_pred, edgecolor='k', alpha=0.7)

plt.plot([y.min(), y.max()], [y.min(), y.max()])

plt.xlabel("Actual Sale Price")

plt.ylabel("Predicted Sale Price")

plt.title(f"Actual vs Predicted Sale Price (Accuracy: {r2*100:.2f}%)")

plt.grid(True)

plt.show()
```

```
---- Model Performance Metrics ----
Mean Absolute Error (MAE): 10126.1733
Mean Squared Error (MSE): 144540866.0004
Root Mean Squared Error (RMSE): 12022.5150
R-squared Score (R2): 0.9770 (97.70%)
Mean Absolute Percentage Error (MAPE): 0.0470 (4.70%)

---- Regression Equation Details ----
Intercept (b0): -1555960.9020
               Coefficient
Id                3.695564
OverallQual   12033.078106
GrLivArea        55.247721
GarageCars    19399.549319
TotalBsmtSF      25.489980
YearBuilt       767.411834
FullBath      10746.959166
BedroomAbvGr   4968.399174
LotArea          -0.203670
```

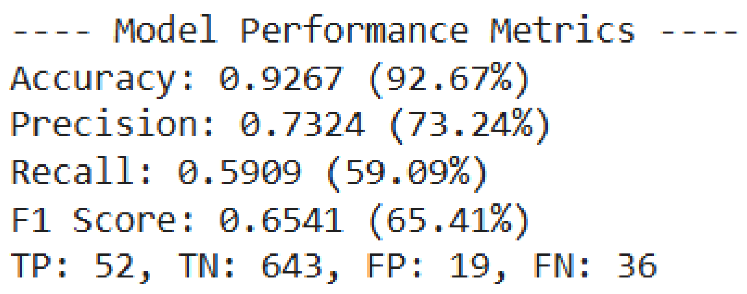


```
from google.colab import files

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


uploaded = files.upload()


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


target = df.columns[-1]

if df[target].dtype == object:

    df[target] = df[target].astype('category').cat.codes


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

y = df[target]


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


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)


scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)
```

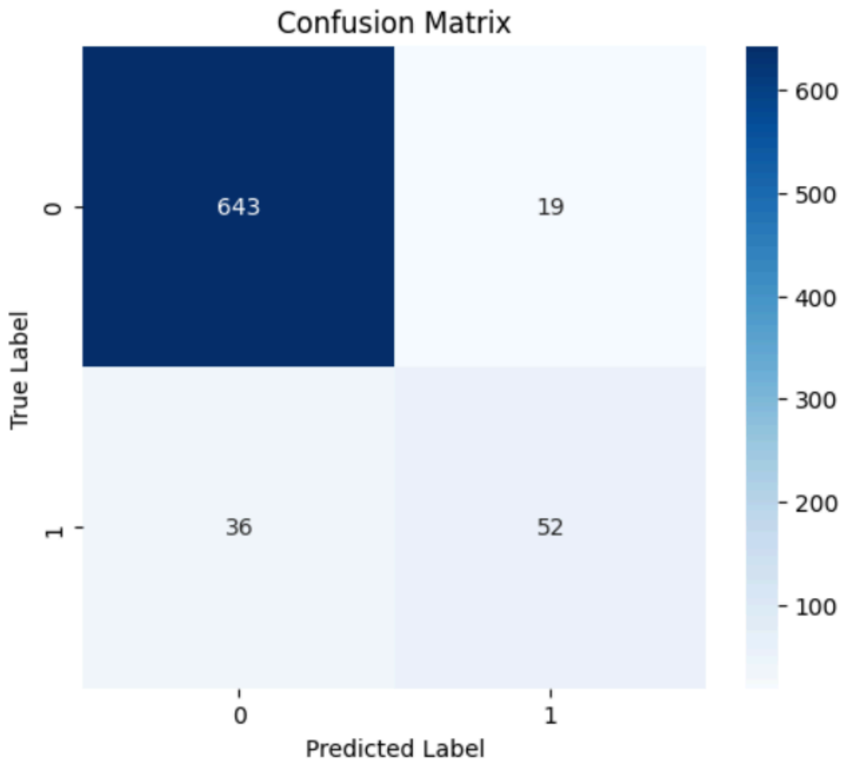
```
model                                 =
LogisticRegression(max_iter=1
0000)

model.fit(X_train, y_train)


y_pred                                =
model.predict(X_test)


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("---- Model Performance
Metrics ----")

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

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

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

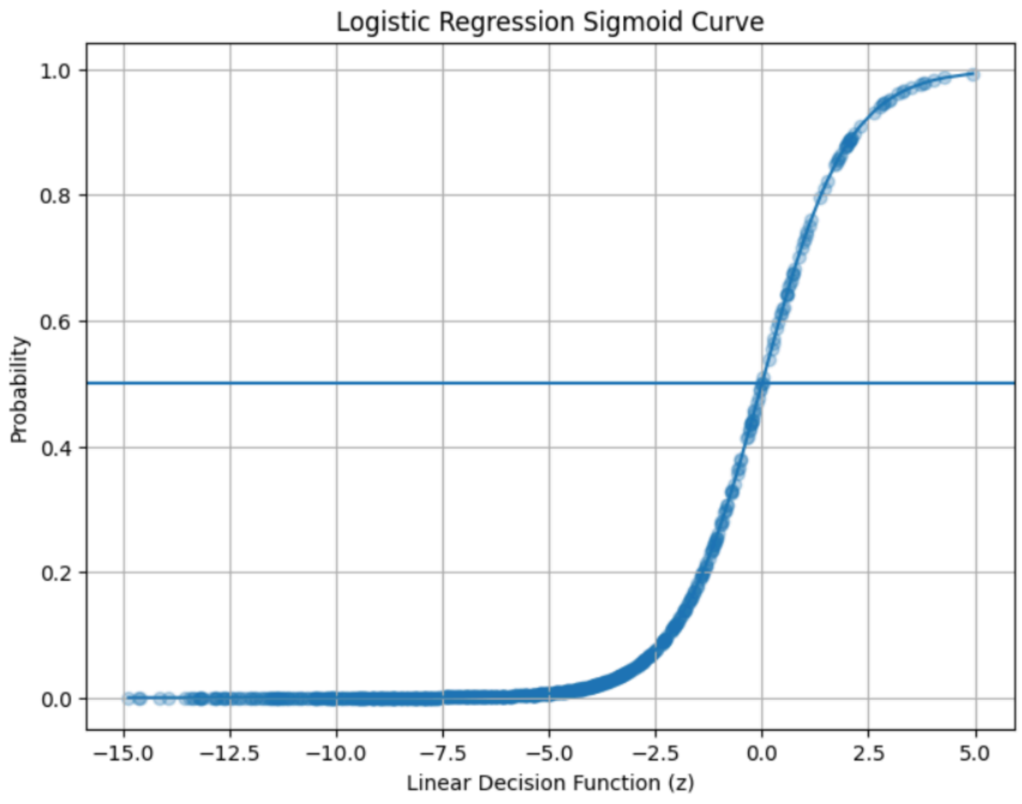
```
print(f"F1 Score: {f1:.4f}
({f1*100:.2f}%)")


cm = confusion_matrix(y_test,
y_pred)

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

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


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()


z                                     =
model.decision_function(X_tes
t)

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()
```



```
---- Model Performance Metrics ----
Accuracy: 0.9267 (92.67%)
Precision: 0.7324 (73.24%)
Recall: 0.5909 (59.09%)
F1 Score: 0.6541 (65.41%)
TP: 52, TN: 643, FP: 19, FN: 36
```


Confusion Matrix
643
19
36
52
True Label
Predicted Label
0
1
0
1
600
500
400
300
200
100


Logistic Regression Sigmoid Curve
Probability
Linear Decision Function (z)
1.0
0.8
0.6
0.4
0.2
0.0
−15.0
−12.5
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0

Conclusion:

This experiment demonstrated the practical implementation of Linear Regression for house price prediction and Logistic Regression for rainfall 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.