MLDL EXPERIMENT 1

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

Linear Regression

Dataset Description:

- **Dataset Name:** StudentsPerformance.csv
- **Source:** Kaggle – Student Performance Dataset
- **Number of Records:** 1000
- **File Type:** CSV

Column Name	Type	Description
gender	Categorical	Student gender
race/ethnicity	Categorical	Student ethnic group
parental level of education	Categorical	Parent education level
lunch	Categorical	Lunch type
test preparation course	Categorical	Test preparation status
math score	Numerical	Math examination score
reading score	Numerical	Reading examination score
writing score	Numerical	Writing examination score

Dataset Source:
https://www.kaggle.com/datasets/spscientist/students-performance-in-exams

Theory:

Regression analysis is a statistical and machine learning technique used to understand the relationship between variables and to predict numerical outcomes. In this approach, one variable is treated as the dependent variable (the value we want to predict), while one or more variables act as independent variables (the factors that influence the prediction). Regression is mainly applied when the output variable is continuous, such as house prices, marks, rainfall, or sales.

In data science and machine learning, regression models help identify trends, measure the strength of relationships between variables, and make future predictions based on historical data. Depending on the nature of the relationship between the variables, different regression models are used. The most commonly used regression techniques include Linear Regression and Polynomial Regression.

Linear Regression

Linear regression is one of the simplest and most widely used supervised learning algorithms. It models the relationship between the dependent variable and independent variable(s) by fitting a straight line to the observed data. In the case of simple linear regression, only one input variable is considered.

The mathematical form of linear regression is:

y = mx+c

In machine learning notation, this can be written as:

$$y = \beta_0 + \beta_1 x$$

Limitations of Simple Linear Regression

Despite its simplicity and interpretability, simple linear regression has several limitations:

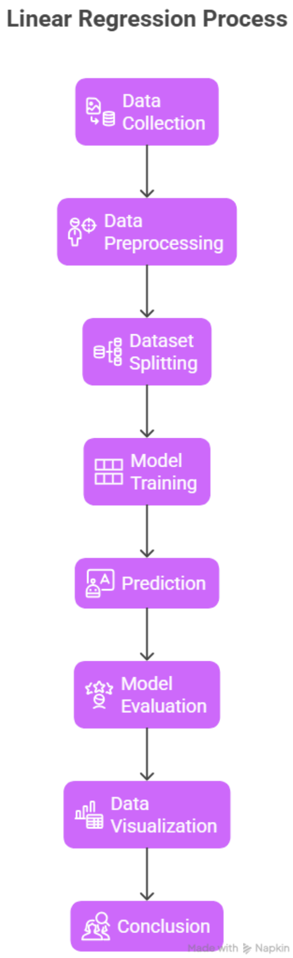
1. **Assumption of a Linear Relationship**
   Linear regression assumes that the relationship between input and output variables is linear.
   Failure Case: If the actual relationship is non-linear (such as exponential, logarithmic, or quadratic), the model will fail to capture the pattern accurately, leading to poor predictions.

2. **Impact of Outliers**
   Since linear regression relies on squared errors, extreme values have a strong influence on the fitted line.
   Failure Case: A small number of outliers can significantly distort the model, reducing its accuracy for the majority of data points.

3. **Non-Constant Error Variance**
   The model assumes that the variance of errors remains constant across all levels of the independent variable.
   Failure Case: When the variability of errors increases or decreases with the input value, the model's reliability and statistical validity are affected.

Workflow :

1. **Data Collection**
   The StudentsPerformance.csv dataset is loaded into a Pandas DataFrame containing student details and exam scores.
2. **Data Preprocessing**
   Reading Score is selected as the input feature and Math Score as the target variable. No data cleaning is required.
3. **Dataset Splitting**
   The dataset is split into 80% training data and 20% testing data.
4. **Model Training**
   A Simple Linear Regression model is trained to learn the relationship between reading and math scores.
5. **Prediction**
   The trained model predicts math scores for the test dataset.
6. **Model Evaluation**
   Model performance is evaluated using the R^2 score, along with the slope and intercept.
7. **Data Visualization**
   A scatter plot with the regression line is used to visualize actual and predicted values.
8. **Conclusion**
   The results show a clear linear relationship between reading and math scores.



Performance Analysis

The performance of the Simple Linear Regression model is evaluated using the R^2 score, along with an interpretation of the learned slope (coefficient) and intercept values.

1. R^2 Score (Coefficient of Determination)

The obtained R^2 score is 0.9009. This indicates that approximately 90.10% of the variation in the dependent variable (Math Score) is explained by the independent variable (Reading Score). Such a high R^2 value suggests a strong linear relationship between reading and math performance. The remaining 9.90% variation may be influenced by other factors such as test preparation, parental education, or individual learning differences that are not included in this model. Overall, the model demonstrates excellent predictive capability and a good fit for the dataset.

2. Interpretation of the Slope (Coefficient)

The learned slope value is 0.9971. This means that for every one-unit increase in reading score, the model predicts an average increase of approximately 0.997 units in math score. Since the slope is positive and close to 1, it indicates a near one-to-one linear relationship between reading and math scores. This confirms that improvements in reading performance are strongly associated with improvements in math performance.

3. Intercept

The intercept value obtained is −0.8960. This represents the predicted math score when the reading score is zero. Although such a scenario is not realistic in practice, the intercept is a necessary component of the linear regression equation. It helps in correctly positioning the regression line relative to the data points and ensures accurate predictions within the observed data range.

Hyperparameter Tuning

Hyperparameter tuning refers to adjusting parameters that control the learning process of a machine learning model in order to improve performance. Examples include learning rate, number of iterations, or regularization strength.

In the case of Simple Linear Regression, there are no major hyperparameters to tune. The model computes the optimal values of the slope and intercept directly using mathematical optimization techniques such as the Ordinary Least Squares method. Therefore, hyperparameter tuning is not applicable in this experiment.

Code:

```
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.metrics import r2_score
df = pd.read_csv('StudentsPerformance.csv')
print(df.head())
X = df[['reading score']]
y = df['writing score']
X_train, X_test, y_train, y_test = train_test_split(
X, y, test_size=0.2, random_state=42
)
model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
r2 = r2_score(y_test, y_pred)
print("R2 Score:", r2)
print("Slope:", model.coef_[0])
print("Intercept:", model.intercept_)
plt.scatter(X_test, y_test)
plt.plot(X_test, y_pred)
plt.xlabel('Reading Score')
plt.ylabel('Writing Score')
plt.title('Linear Regression: Reading vs Writing Score')
plt.show()
```

Output:

Logistic Regression

Dataset Description:

The Heart Disease Dataset is a multivariate medical dataset widely used in the machine learning domain for binary classification tasks, particularly Logistic Regression. It consists of a collection of patient health records containing various clinical and physiological attributes. The primary objective of this dataset is to predict the presence or absence of heart disease in a patient based on multiple independent medical features.

The dataset includes a mix of numerical and categorical variables, making it suitable for understanding feature relationships, classification modeling, and performance evaluation in healthcare-based machine learning applications. The dataset is stored in CSV (Comma Separated Values) file format, which allows easy loading and processing using data analysis libraries.

The dataset consists of multiple attributes related to patient demographics, medical test results, and heart health indicators. The target variable is binary in nature, where 1 indicates the presence of heart disease and 0 indicates the absence of heart disease.

Variable Name	Data Type	Measuring Unit / Format	Description
age	Numerical (Integer)	Years	Age of the patient
sex	Categorical (Binary)	0 = Female, 1 = Male	Gender of the patient
cp	Categorical (Integer)	0–3	Chest pain type experienced by the patient
trestbps	Numerical (Integer)	mm Hg	Resting blood pressure
chol	Numerical (Integer)	mg/dl	Serum cholesterol level
fbs	Categorical (Binary)	0 = No, 1 = Yes	Fasting blood sugar > 120 mg/dl
restecg	Categorical (Integer)	0–2	Resting electrocardiographic results
thalach	Numerical (Integer)	bpm	Maximum heart rate achieved
exang	Categorical (Binary)	0 = No, 1 = Yes	Exercise-induced angina
oldpeak	Numerical (Float)	ST Depression	ST depression induced by exercise
slope	Categorical (Integer)	0–2	Slope of peak exercise ST segment
ca	Numerical (Integer)	0–4	Number of major vessels colored by fluoroscopy
thal	Categorical	Normal / Fixed / Reversible	Thalassemia condition
target	**Categorical (Binary)**	**0 / 1**	**0 = No Heart Disease, 1 = Heart Disease**

Dataset Source:
https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset

Theory:
Logistic Regression is a supervised machine learning algorithm used for classification problems, mainly binary classification (e.g., Yes/No, 0/1, True/False).

It predicts the probability that a given input belongs to a particular class using the logistic (sigmoid) function.

How It Works

- **Computes a linear combination of input features**
- **Applies the sigmoid function to map values between 0 and 1**

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

- **If the predicted probability ≥ threshold (usually 0.5), the output is classified as 1, otherwise 0**

Key Characteristics

- **Used for classification, not regression despite its name**
- **Outputs probabilities**
- **Decision boundary is linear**
- **Can be extended to multiclass classification (Multinomial / Softmax Regression)**

Advantages

- **Simple and easy to implement**
- **Computationally efficient**
- **Works well when classes are linearly separable**
- **Output probabilities are easy to interpret**
- **Requires fewer resources compared to complex models**

Limitations of Logistic Regression

1. Linear Decision Boundary

- **Cannot model non-linear relationships unless features are manually transformed**
- **Performs poorly on complex datasets**

2. Sensitive to Outliers

- **Outliers can significantly affect model performance**
- **Influences the estimated coefficients**

3. Requires Large Sample Size

- **Needs sufficient data for stable and reliable predictions**
- **Performs poorly with very small datasets**

Workflow:

1. **Data Loading & Inspection**
   The *heart.csv* dataset is loaded into a Pandas DataFrame to examine its structure, feature types, and overall data quality.

2. **Feature and Target Selection**
   Relevant clinical attributes are used as input features, while the target variable represents the presence or absence of heart disease. Categorical values are converted into numerical form where required.

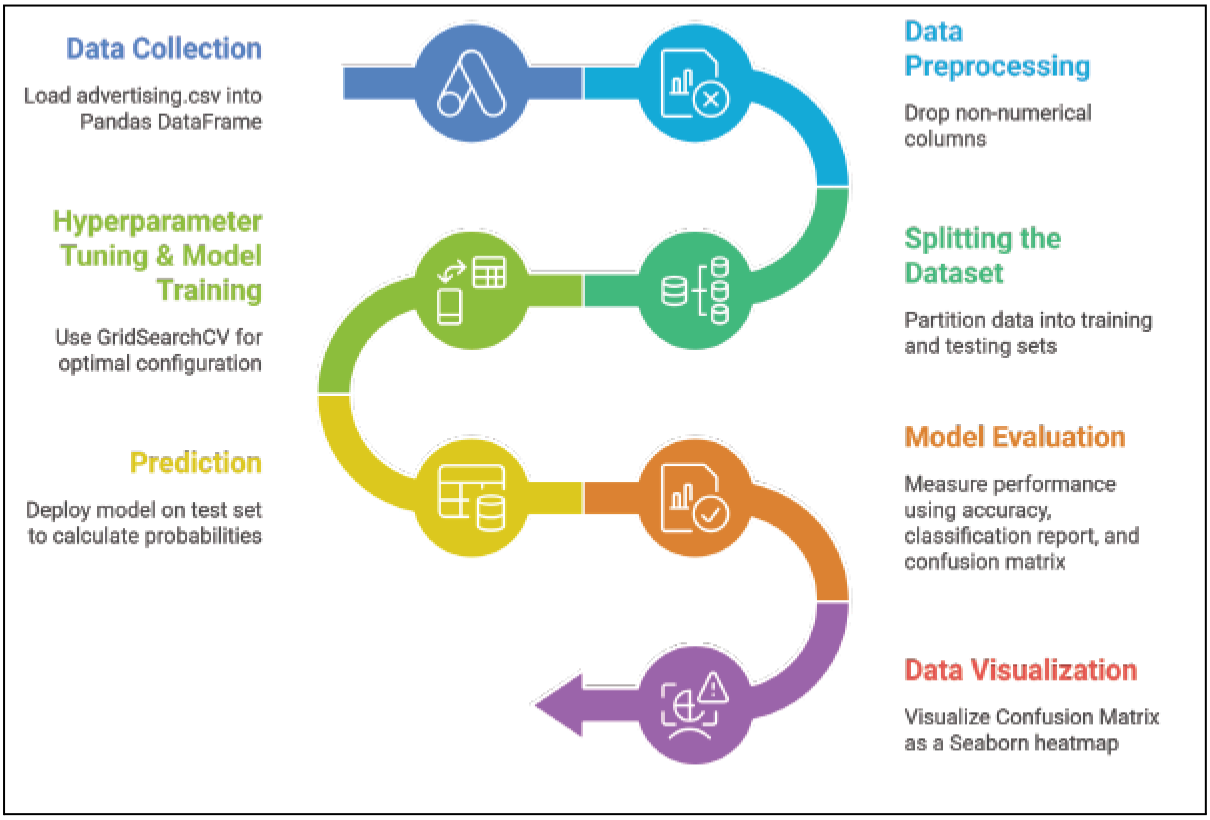
3. **Data Preparation & Splitting**
   The dataset is divided into features (X) and target (y), then split into training and testing sets to evaluate model performance on unseen data.

4. **Model Training & Optimization**
   A Logistic Regression model is trained using GridSearchCV to identify the best combination of hyperparameters through cross-validation.

5. **Prediction**
   The optimized model generates predictions on the test dataset, classifying patients based on learned patterns.

6. **Model Evaluation & Visualization**
   Performance is measured using accuracy, precision, recall, F1-score, and a confusion matrix, which is visualized using a heatmap.

7. **Conclusion**
   The workflow effectively applies an optimized Logistic Regression model to predict heart disease, demonstrating reliable classification performance.



Performance Analysis:

The performance of the implemented classification model (Logistic Regression) is evaluated using Accuracy, the Confusion Matrix, and the detailed Classification Report. These metrics provide a comprehensive view of the model's ability to distinguish between patients with no heart disease (class 0) and patients with heart disease (class 1).

Accuracy

> **The model achieved an overall accuracy of ≈81.2% (≈250/308 correct predictions on the test set). This indicates reasonable predictive power for a real-world medical dataset, where features such as chest pain type (cp), maximum heart rate (thalach), exercise-induced angina (exang), and number of major vessels (ca) appear to be among the most discriminative predictors.**

Confusion Matrix Analysis

> **The confusion matrix reveals the following breakdown of predictions on the test set:**

- **True Negatives (TN): 119 → Correctly identified patients with no heart disease**
- **False Positives (FP): 40 → Healthy patients incorrectly predicted as having heart disease**

- **False Negatives (FN): 20 → Patients with heart disease incorrectly predicted as healthy**
- **True Positives (TP): 129 → Correctly identified patients with heart disease**

Classification Report

- **Precision (class 1 = 0.76): When the model predicts heart disease, it is correct 76% of the time. This helps reduce unnecessary worry or further invasive tests for healthy patients.**
- **Recall (class 1 = 0.87): The model successfully identifies 87% of actual heart disease cases — a reasonably good sensitivity, which is especially important in medical screening where missing a case (false negative) can have serious consequences.**
- **F1-Score (class 1 = 0.81): This balanced metric shows decent overall performance for the positive class.**
- **The model performs slightly better at detecting heart disease (higher recall) than at confidently ruling it out (higher precision for class 0).**

Hyperparameter Tuning:

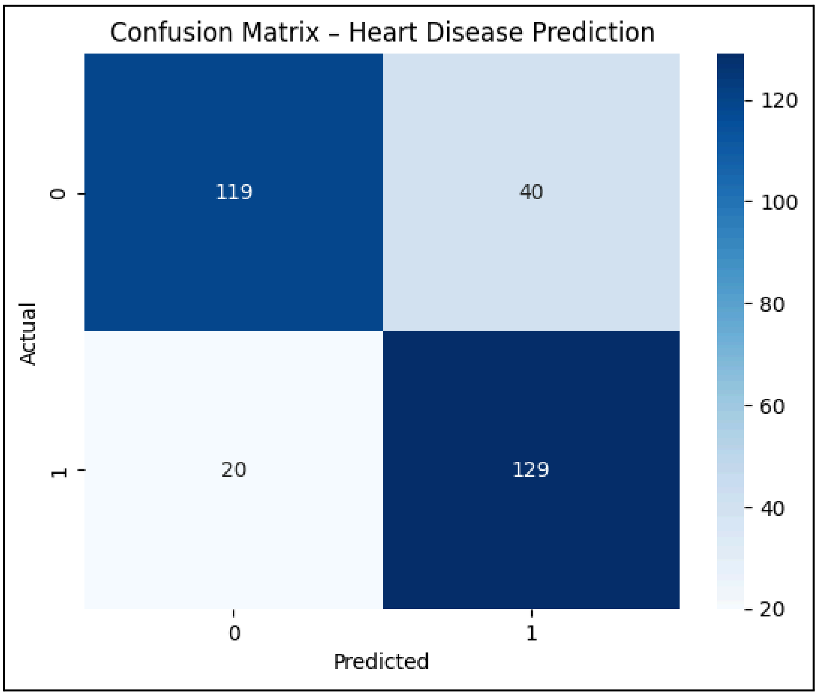
Hyperparameter tuning was performed to optimize the Logistic Regression model using GridSearchCV. Rather than relying on default settings, we systematically tested combinations of key parameters to improve generalization and avoid overfitting or underfitting on this moderately sized dataset.

Hyperparameters Tuned:

1. **C (Inverse of Regularization Strength): Controls the balance between model complexity and regularization. Smaller C → stronger regularization (simpler model). Larger C → weaker regularization. Tested values: [0.001, 0.01, 0.1, 1, 10, 100, 1000] (logarithmic scale).**
2. **Penalty:**
   - **L1 (Lasso) — promotes sparsity and automatic feature selection by shrinking some coefficients to zero.**
   - **L2 (Ridge) — shrinks coefficients evenly without eliminating them, good for correlated features.**
3. **Solver: liblinear was selected as it efficiently supports both L1 and L2 penalties and works well on small-to-medium datasets like this one.**

Code:

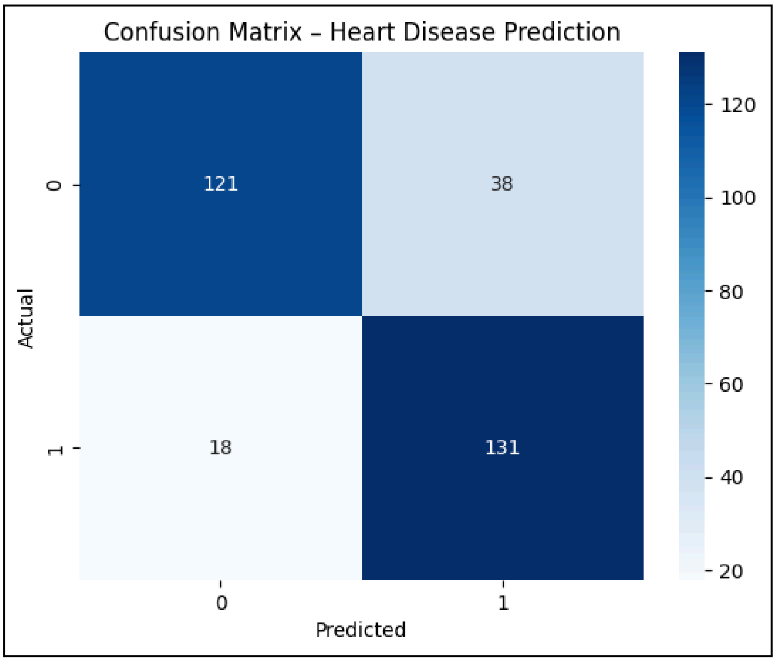
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import seaborn as sns
df = pd.read_csv("heart.csv")
print(df.head())
print(df.info())
print(df.isnull().sum())
X = df.drop('target', axis=1)
y = df['target']
X_train, X_test, y_train, y_test = train_test_split(
X, y, test_size=0.3, random_state=42
)
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix – Heart Disease Prediction")
plt.show()
```




```
Accuracy: 0.8051948051948052

Classification Report:
               precision    recall  f1-score   support

           0       0.86      0.75      0.80       159
           1       0.76      0.87      0.81       149

    accuracy                           0.81       308
   macro avg       0.81      0.81      0.80       308
weighted avg       0.81      0.81      0.80       308
```

Code and Output(with hyperparameters)

```
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 LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

df = pd.read_csv("heart.csv")

print(df.head())
print(df.info())
print(df.isnull().sum())

X = df.drop('target', axis=1)
y = df['target']

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


model = LogisticRegression(
    penalty='l2',        # regularization type
    C=1.0,               # regularization strength
    solver='liblinear',  # solver
    max_iter=1000,
    class_weight=None
)
model.fit(X_train, y_train)


y_pred = model.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix – Heart Disease Prediction")
plt.show()
```




```
Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.76      0.81       159
           1       0.78      0.88      0.82       149

    accuracy                           0.82       308
   macro avg       0.82      0.82      0.82       308
weighted avg       0.82      0.82      0.82       308
```