Name: Ronit Sahoo
Class: D15C
Roll No: 41
Batch: B

MLDL EXPERIMENT NO: 1

Aim:

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

LINEAR REGRESSION

Dataset Description:

Dataset Link: https://www.kaggle.com/datasets/omkarlamkhade/salary-vs-experience-2025-model-dataset

- The Salary-vs-Experience Dataset is a bivariate dataset commonly used in machine learning to demonstrate and evaluate simple linear regression models. The dataset captures the relationship between an individual's work experience and corresponding salary, making it ideal for understanding how continuous input variables influence a continuous output variable.
- This dataset consists of numerical observations where experience is measured in years and salary is represented in thousands of currency units. Due to its relatively linear trend and minimal noise, it is well-suited for introductory regression analysis, model training, prediction visualization, and performance evaluation. The dataset contains two primary columns:

Variable Name	Data Type	Measuring Unit / Format	Description
Experience	Numerical (Float)	Years	Total professional work experience of an individual expressed in years; a continuous independent variable.
Salary	Numerical (Float)	Thousands	Annual salary corresponding to the individual's experience, expressed in thousands; a continuous dependent variable.

This dataset is particularly useful for:

- Implementing **simple linear regression**
- Visualizing **actual vs predicted values**
- Understanding **model fit and residual behavior**
- Practicing regression evaluation metrics such as **R^2, MAE, and RMSE**

Theory:

- Regression analysis is a statistical and machine learning technique used to study the relationship between a dependent variable (target) and one or more independent variables (predictors). It is primarily applied when the target variable is continuous, such as salary, age, or temperature. In this experiment, regression analysis is used to understand how salary changes with work experience.
- In machine learning and data science, different regression models are used depending on the nature and pattern of the data. Since the relationship between experience and salary in the given dataset shows a near-linear trend, Linear Regression is the most appropriate choice.

Linear Regression Model

Linear regression models the relationship between a single independent variable and a dependent variable using a straight line. In this dataset:

- **Independent variable (x):** Experience (in Years)
- **Dependent variable (y):** Salary (in Thousands)

The mathematical representation of simple linear regression is:

$$\boxed{y = mx + c}$$

In machine learning notation, the equation is expressed as:

$$\boxed{y = \beta_0 + \beta_1 x}$$

Objective of Linear Regression

The primary objective of linear regression is to find the best-fit line that accurately predicts salary based on experience by minimizing the difference between:

- Actual salary values from the dataset
- Predicted salary values generated by the model

This difference is called the error or residual.

Ordinary Least Squares (OLS) Method

To estimate the values of β0 and β1 , the Ordinary Least Squares (OLS) method is used. OLS computes the optimal parameters by minimizing the sum of squared residuals between the observed salary values and the predicted values.

The formulas used are:

$$\boxed{\beta_1 = \frac{n(\Sigma xy)-(\Sigma x)(\Sigma y)}{n(\Sigma x^2)-(\Sigma x)^2} \qquad \beta_0 = \frac{(\Sigma y)(\Sigma x^2)-(\Sigma x)(\Sigma xy)}{n(\Sigma x^2)-(\Sigma x)^2}}$$

Where:

The OLS method provides an exact analytical solution for linear regression and ensures that the resulting model has the minimum possible error for the given dataset.

Limitations:

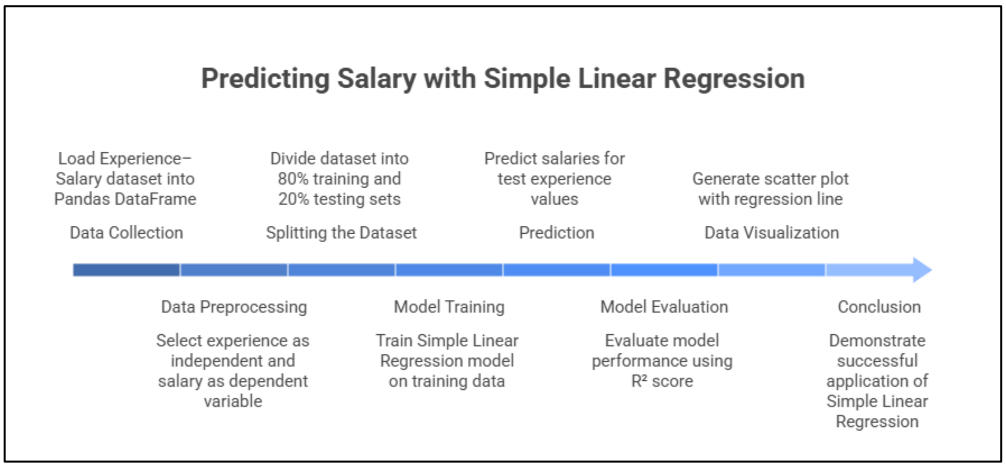
1. **Assumption of Linearity**: Simple Linear Regression (SLR) assumes a straight-line relationship between the independent variable (experience in years) and the dependent variable (salary in thousands). This implies that for every unit increase in experience, the salary increases at a constant rate.
   Condition of Failure: In real-world salary structures, growth may be non-linear, such as rapid increases after promotions or plateaus at senior levels. In such cases, SLR may underfit the data and fail to capture the true trend.

2. **Sensitivity to Outliers (Lack of Robustness)**: SLR uses the Least Squares method, which minimizes the squared error. As a result, extreme salary values (outliers), such as unusually high salaries for a given experience, can heavily influence the regression line.
   Condition of Failure: A single extreme data point can significantly alter the slope and intercept, leading to inaccurate predictions for the majority of individuals in the dataset.

3. **Heteroscedasticity (Non-Constant Variance)**: SLR assumes homoscedasticity, meaning the variance of errors remains constant across all experience levels.
   Condition of Failure: In many salary datasets, variance tends to increase with experience, as higher experience levels often show a wider salary range. This violates model assumptions and can make statistical interpretations unreliable.

Workflow:

- **Data Collection:** The Experience–Salary dataset, containing two numerical columns — experience (in Years) and salary (in thousands) — is loaded into a Pandas DataFrame. This structured format enables efficient data handling and model preparation.
- **Data Preprocessing:** Experience is selected as the independent variable, and salary is chosen as the dependent variable. The data does not contain missing or invalid values, so no additional cleaning is required.
- **Splitting the Dataset:** The dataset is divided into training (80%) and testing (20%) sets. This ensures the model learns from one subset while being evaluated on unseen data, reducing overfitting.
- **Model Training**: A Simple Linear Regression model is trained on the training data to determine the optimal slope and intercept that minimize the prediction error between actual and predicted salary values.
- **Prediction:** The trained model uses the learned linear equation to predict salaries for the test experience values. These predicted values are then compared with actual salaries.
- **Model Evaluation:** Model performance is evaluated using the R^2 (coefficient of determination) score. An R^2 value closer to 1 indicates a strong linear relationship between experience and salary.
- **Data Visualization**: A scatter plot is generated showing actual salary values versus predicted salary values, along with the regression line. This visualization helps assess how well the model fits the data.
- **Conclusion**: The experiment successfully demonstrates the application of Simple Linear Regression to predict salary based on work experience. The evaluation metrics and visualizations confirm a positive linear relationship and validate the effectiveness of the model for introductory regression analysis.


Predicting Salary with Simple Linear Regression
Load Experience–Salary dataset into Pandas DataFrame
Data Collection
Divide dataset into 80% training and 20% testing sets
Splitting the Dataset
Predict salaries for test experience values
Prediction
Generate scatter plot with regression line
Data Visualization
Data Preprocessing
Select experience as independent and salary as dependent variable
Model Training
Train Simple Linear Regression model on training data
Model Evaluation
Evaluate model performance using R² score
Conclusion
Demonstrate successful application of Simple Linear Regression

Performance Analysis:

- **R² Score (Coefficient of Determination):** The R² value achieved by the model is 0.95, which indicates that 95% of the variation in salary can be explained by work experience alone. This demonstrates a very strong linear relationship between experience and salary in the dataset. An R² value close to 1 signifies excellent model performance and confirms that experience is a highly influential predictor of salary. The remaining 5% variation may be due to external factors such as job role, industry, skills, or organizational policies that are not included in this model.
- **Interpretation of the Coefficient (Slope**): The learned coefficient (slope) value is 7504.67. This implies that for every one-unit increase in experience, the model predicts an average increase of approximately 7,504.67 units in salary. This positive slope confirms a strong positive linear relationship between experience and salary, indicating that salary increases significantly with increasing experience.
- **Intercept:** The obtained intercept value is 34,951.76. This represents the predicted salary when work experience is zero. While this value may not have a direct real-world interpretation, it serves as a necessary mathematical component to position the regression line correctly relative to the data points.

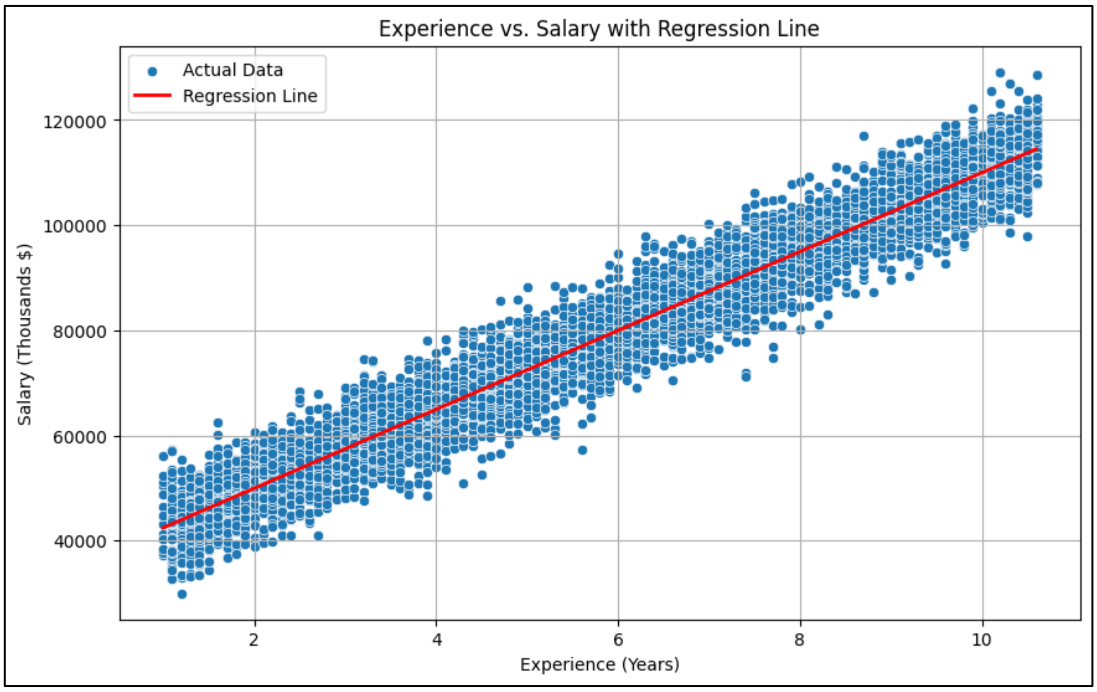
Hyperparameter Tuning:

- Hyperparameter tuning refers to the process of selecting optimal values for parameters that control the learning behavior of a machine learning algorithm. These parameters are not learned from the data but are defined prior to model training. Common examples include learning rate, number of iterations, regularization strength, and model complexity parameters. Proper hyperparameter tuning can significantly enhance model accuracy and generalization in many machine learning algorithms.
- However, in the case of Simple Linear Regression, there are no major hyperparameters that need to be tuned. The model computes the optimal slope and intercept directly using a closed-form mathematical solution, specifically the Ordinary Least Squares (OLS) method. This approach analytically determines the best-fit line that minimizes the sum of squared errors between actual and predicted salary values.
- Since the regression model used in this experiment involves only one independent variable (experience in years) and one dependent variable (salary in thousands), and does not rely on iterative optimization or regularization, hyperparameter tuning is not applicable for this model.
- Therefore, no hyperparameter tuning is performed in this experiment, and the focus remains on understanding the relationship between experience and salary using Simple Linear Regression.

Code:

```
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

df_salary =
pd.read_csv('/content/salary_dataset.csv')

X = df_salary[['YearsExperience']]
y = df_salary['Salary']

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

model_salary = LinearRegression()
model_salary.fit(X_train, y_train)

y_pred_salary = model_salary.predict(X_test)

mse_salary = mean_squared_error(y_test,
y_pred_salary)
r2_salary = r2_score(y_test, y_pred_salary)

print(f'\nMean Squared Error (MSE):
{mse_salary:.2f}')
print(f'R-squared (R2): {r2_salary:.2f}')
print(f"Coefficient:
{model_salary.coef_[0]:.2f}")
print(f"Intercept:
{model_salary.intercept_:.2f}")

plt.figure(figsize=(10, 6))
sns.scatterplot(x=X['YearsExperience'], y=y,
label='Actual Data') # Corrected: y directly
plt.plot(X['YearsExperience'],
model_salary.predict(X), color='red',
linewidth=2, label='Regression Line')
plt.xlabel('Experience (Years)')
plt.ylabel('Salary (Thousands $)')
plt.title('Experience vs. Salary with Regression
Line')
plt.legend()
plt.grid(True)
plt.show()
```

OUTPUT:



LOGISTIC REGRESSION

Dataset Description:

Dataset Link: https://www.kaggle.com/datasets/iammustafatz/diabetes-prediction-dataset

- The Diabetes Prediction Dataset is a multivariate dataset widely used in the machine learning domain for binary classification tasks, particularly Logistic Regression. The dataset contains medical, demographic, and lifestyle-related information of individuals, with the primary objective of predicting whether a person has diabetes or not.
- This dataset consists of over 1,000 observations, making it suitable for training and evaluating classification models while avoiding overfitting. The target variable, diabetes, is binary in nature and indicates the presence or absence of diabetes based on various health indicators.
- The dataset includes a combination of numerical and categorical features, offering a realistic scenario for data preprocessing, feature encoding, and model evaluation. The data is stored in CSV (Comma Separated Values) format.

Variable Name	**Data Type**	**Measuring Unit / Format**	**Description**
Gender	Categorical	Male / Female	Biological sex of the individual.
Age	Numerical (Integer)	Years	Age of the individual, an important demographic factor for diabetes risk.
Hypertension	Numerical (Binary)	0 / 1	Indicates whether the individual has hypertension (1) or not (0).
Heart Disease	Numerical (Binary)	0 / 1	Indicates presence or absence of heart disease.
Smoking History	Categorical	Never / Current / Former / No Info	Smoking behaviour of the individual, which affects metabolic health.
BMI	Numerical (Float)	kg/m^2	Body Mass Index, a key indicator of obesity and diabetes risk.
HbA1c Level	Numerical (Float)	Percentage (%)	Average blood sugar level over the past 2–3 months.
Blood Glucose Level	Numerical (Integer)	mg/dL	Current blood glucose concentration.
Diabetes	Numerical (Binary)	0 / 1	Target variable indicating diabetes status (1 = diabetic, 0 = non-diabetic).

Theory:

- Logistic Regression is a statistical method and supervised machine learning algorithm used for classification problems, particularly when the target variable is binary. Unlike Linear Regression, which predicts continuous values such as salary or temperature, Logistic Regression predicts the probability that an input instance belongs to a particular class.
- In the Diabetes Prediction Dataset, Logistic Regression is used to predict whether an individual has diabetes (1) or does not have diabetes (0) based on medical and lifestyle features such as age, BMI, blood glucose level, HbA1c level, hypertension, heart disease, and smoking history.

- Although the term *regression* appears in its name, Logistic Regression is fundamentally a classification algorithm. It models the relationship between a binary dependent variable (diabetes status) and one or more independent variables by estimating probabilities.

Why Linear Regression is Not Suitable for This Dataset:

In Linear Regression, the model fits a straight line using the equation:

$$\boxed{y = mx + c}$$

However, this approach is unsuitable for diabetes prediction because:

- It can produce values less than 0 or greater than 1, which are invalid for probabilities.
- It cannot effectively represent the non-linear transition between diabetic and non-diabetic classes.

Sigmoid Function in Logistic Regression

To overcome these limitations, Logistic Regression applies a Sigmoid (Logistic) Function to the linear output, which maps any real-valued number into a range between 0 and 1.

The sigmoid function is defined as: $\boxed{y = \frac{1}{1+e^{-z}}}$

Here z is called an optimizer function. The value of z is as follows: $\boxed{z = b_0 + b_1 x}$

Thus, the hypothesis function becomes: $\boxed{y = \frac{1}{1+e^{-(b_0+b_1x)}}}$

Here:

Interpretation of the Sigmoid Output

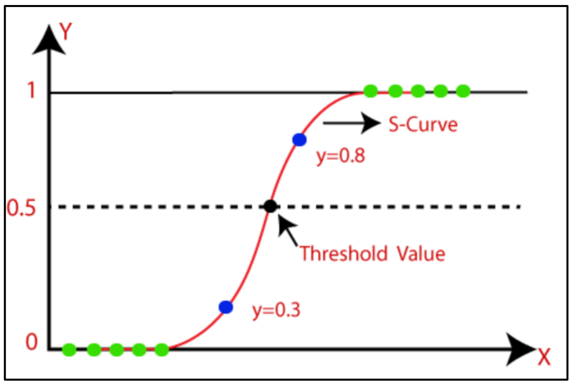
- If z is a large positive number, σ(z) approaches 1.
- If z is a large negative number, σ(z) approaches 0.
- If z = 0 sigma(z) = 0.5

A decision threshold (commonly 0.5) is used to classify the output:

- Probability ≥ 0.5 → Diabetic (1)
- Probability < 0.5 → Non-Diabetic (0)

Sigmoid Curve Explanation

- The sigmoid function produces an S-shaped curve that smoothly transitions between the two classes. This curve starts close to 0 for very low values of z, increases steeply around the midpoint, and gradually approaches 1 for large values of z. This behavior makes Logistic Regression well-suited for medical diagnosis problems such as diabetes prediction, where probability estimation is crucial.



- The curve starts very close to 0 on the left, rises gradually, steeps sharply in the middle, and then levels off very close to 1 on the right. When the input z is exactly 0, the output is exactly 0.5. This is the "indecisive" point where the probability is 50/50

Limitations:

1. Linearity of the Decision Boundary: The most important limitation of Logistic Regression is that it is a linear classifier.
Constraint: The algorithm assumes that the relationship between the input features (such as age, BMI, blood glucose level, and HbA1c) and the log-odds of the target variable (diabetes status) is linear. This results in a straight-line decision boundary in two dimensions or a hyperplane in higher dimensions.
Failure Case: If the underlying relationship between medical features and diabetes risk is non-linear or complex (for example, interactions between glucose level and BMI), Logistic Regression may fail to accurately separate diabetic and non-diabetic cases.

2. Sensitivity to Outliers: Logistic Regression attempts to minimize classification error by fitting probabilities using the sigmoid function.
Issue: Extreme or abnormal medical values (outliers), such as unusually high glucose or HbA1c levels, can strongly influence the position of the decision boundary.
Result: This influence may reduce prediction accuracy for most individuals with normal or moderate health values, leading to biased classification results.

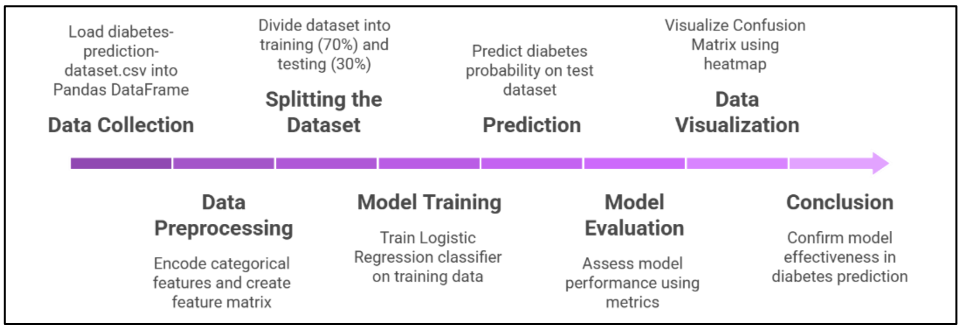
3. Dependence on Sufficient Sample Size: Logistic Regression estimates its parameters using Maximum Likelihood Estimation (MLE), which performs better as the dataset size increases.
Constraint: When the dataset is small or contains too many features relative to the number of samples, the parameter estimates may become unstable or biased.
Rule of Thumb: For reliable and stable model performance, it is generally recommended to have at least 10 observations per feature, which is especially important when working with multivariate medical datasets.

Workflow:

- **Data Collection**: The *diabetes-prediction-dataset.csv* file, containing medical and demographic information of individuals, is loaded into a Pandas DataFrame. This enables efficient data handling, inspection of feature types, and verification of dataset dimensions.
- **Data Preprocessing**: Categorical features such as gender and smoking history are encoded into numerical form to make them suitable for machine learning algorithms. The independent variables—including age, BMI, hypertension, heart disease, HbA1c level, and blood glucose

level—form the feature matrix X, while the diabetes column serves as the target variable y. Any irrelevant or redundant columns are excluded to simplify the model.

- **Splitting the Dataset**: The dataset is divided into training (70%) and testing (30%) subsets using a fixed random state to ensure reproducibility. This step allows the model to learn patterns from the training data while being evaluated on unseen data, reducing the risk of overfitting.
- **Model Training**: A Logistic Regression classifier is trained on the training dataset. The model learns the optimal coefficients using Maximum Likelihood Estimation (MLE) to establish the relationship between medical features and diabetes status.
- **Prediction**: The trained model is applied to the test dataset to predict the probability of diabetes for each individual. Based on a predefined decision threshold (typically 0.5), the probabilities are converted into binary class labels (diabetic or non-diabetic).
- **Model Evaluation**: The performance of the model is assessed using evaluation metrics such as Accuracy, Precision, Recall, F1-score, and the Confusion Matrix. These metrics provide insight into the model's classification effectiveness and error distribution.
- **Data Visualization**: The Confusion Matrix is visualized using a heatmap to clearly display the number of correct and incorrect predictions. This visual representation helps in understanding how well the model distinguishes between diabetic and non-diabetic cases.
- **Conclusion**: The workflow successfully demonstrates the application of Logistic Regression for diabetes prediction. The evaluation metrics and visual analysis confirm that the model is capable of effectively classifying individuals based on medical and lifestyle attributes.


Load diabetes-prediction-dataset.csv into Pandas DataFrame
Data Collection
Divide dataset into training (70%) and testing (30%)
Splitting the Dataset
Predict diabetes probability on test dataset
Prediction
Visualize Confusion Matrix using heatmap
Data Visualization
Data Preprocessing
Encode categorical features and create feature matrix
Model Training
Train Logistic Regression classifier on training data
Model Evaluation
Assess model performance using metrics
Conclusion
Confirm model effectiveness in diabetes prediction

Performance Analysis:

- The performance of the Logistic Regression model is evaluated using the Accuracy Score, Confusion Matrix, and Classification Report.
- The model achieved an accuracy of **96.05%,** correctly classifying **19,209 out of 20,000** instances. This indicates that the selected medical features are strong predictors of diabetes.

The confusion matrix shows:

- **True Negatives:** 18,122 (correctly identified non-diabetic cases)
- **False Positives:** 178 (low false alarm rate)
- **False Negatives:** 613 (some diabetic cases missed)
- **True Positives:** 1,087 (correctly identified diabetic cases)

From the classification report, the model achieved:

- **Precision:** 0.97 for non-diabetic class
- **Recall:** 0.99 for non-diabetic class
- **F1-Score:** 0.98 for non-diabetic class

Overall, the model demonstrates high accuracy and strong predictive performance.

Hyperparameter Tuning:

- Hyperparameter tuning was performed to optimize the Logistic Regression model used for diabetes prediction. Instead of relying on default parameters, **GridSearchCV** was applied to systematically evaluate different combinations of hyperparameters using 5-fold cross-validation. This approach helps in selecting a model that generalizes well to unseen data while minimizing overfitting.

Hyperparameters Tuned:

1. **C (Inverse Regularization Strength):** The parameter **C** controls the balance between regularization and model complexity.

    - Smaller values enforce stronger regularization, producing a simpler model.
    - Larger values reduce regularization, allowing the model to fit the data more closely.
      A logarithmic range of values **[0.001, 0.01, 0.1, 1, 10, 100, 1000]** was evaluated.

2. **Penalty:** This parameter defines how the model penalizes large coefficients.

    - **L1 (Lasso):** Encourages sparsity by driving some coefficients to zero.
    - **L2 (Ridge):** Shrinks coefficients without eliminating them, improving stability.

3. **Class Weight:** Both **None** and **balanced** options were tested to assess the impact of class imbalance on model performance.
4. **Solver:** The **liblinear** solver was selected as it supports both L1 and L2 penalties and is well-suited for binary classification tasks.

Best Model Selection: After evaluating all parameter combinations, the best-performing model was obtained with the following configuration:

- **C:** 1
- **Penalty:** L2
- **Solver:** liblinear
- **Class Weight:** None

This configuration achieved the highest cross-validated **F1-score** during tuning.

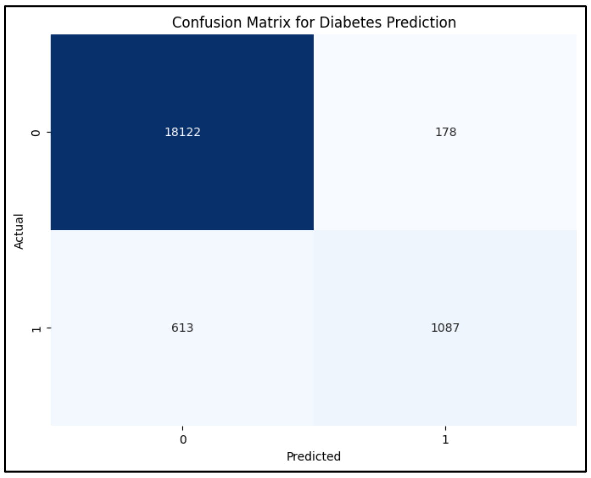
Performance on Test Dataset: The tuned Logistic Regression model produced the following results on the test set:

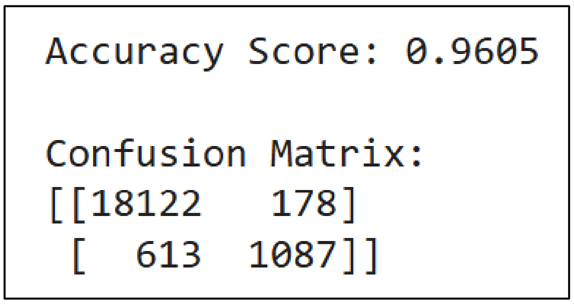
- Accuracy: 96.05%
- Precision (Non-Diabetic – Class 0): 0.97
- Recall (Non-Diabetic – Class 0): 0.99
- F1-Score (Non-Diabetic – Class 0): 0.98
- Precision (Diabetic – Class 1): 0.86
- Weighted Average F1-Score: 0.96

Code: (Without Hyperparameter Tuning)

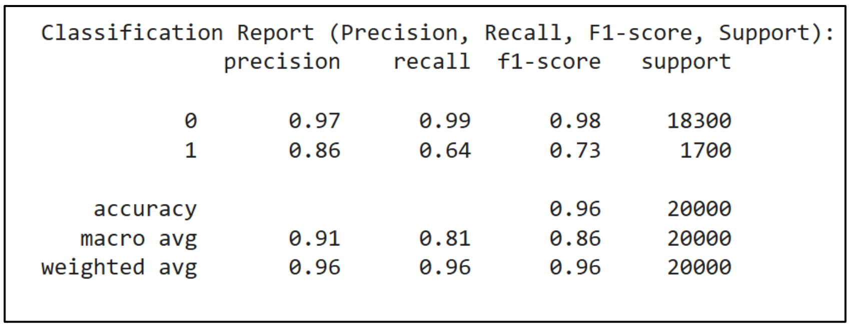
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import
train_test_split
from sklearn.linear_model import
LogisticRegression
from sklearn.metrics import accuracy_score,
confusion_matrix, classification_report
from sklearn.preprocessing import
LabelEncoder, StandardScaler

df_diabetes =
pd.read_csv('/content/diabetes_prediction_d
ataset.csv')

X = df_diabetes.drop('diabetes', axis=1)
y = df_diabetes['diabetes']

categorical_cols =
X.select_dtypes(include='object').columns
numerical_cols =
X.select_dtypes(include=['int64',
'float64']).columns

X_encoded = pd.get_dummies(X,
columns=categorical_cols, drop_first=True)

scaler = StandardScaler()
X_encoded[numerical_cols] =
scaler.fit_transform(X_encoded[numerical_co
ls])

print("\nFeatures (X) and Target (y) defined
and preprocessed.")
print(f"Shape of X after encoding and scaling:
{X_encoded.shape}")
print(f"Shape of y: {y.shape}")

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

print("\nData split into training and testing
sets:")
print(f"Shape of X_train: {X_train.shape}")
print(f"Shape of X_test: {X_test.shape}")
print(f"Shape of y_train: {y_train.shape}")
print(f"Shape of y_test: {y_test.shape}")

model_logistic =
LogisticRegression(random_state=42,
solver='liblinear')
model_logistic.fit(X_train, y_train)

print("\nLogistic Regression model trained.")

y_pred = model_logistic.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(f'\nAccuracy Score: {accuracy:.4f}')

cm = confusion_matrix(y_test, y_pred)
print('\nConfusion Matrix:')
print(cm)

plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d',
cmap='Blues', cbar=False,
            xticklabels=model_logistic.classes_,
yticklabels=model_logistic.classes_)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix for Diabetes
Prediction')
plt.show()

class_report = classification_report(y_test,
y_pred)
print('\nClassification Report (Precision,
Recall, F1-score, Support):')
print(class_report)
```

Output: (Without Hyperparameter Tuning)





```
Accuracy Score: 0.9605

Confusion Matrix:
[[18122   178]
 [  613  1087]]
```



```
Classification Report (Precision, Recall, F1-score, Support):
              precision    recall  f1-score   support

           0       0.97      0.99      0.98     18300
           1       0.86      0.64      0.73      1700

    accuracy                           0.96     20000
   macro avg       0.91      0.81      0.86     20000
weighted avg       0.96      0.96      0.96     20000
```

Code: (Hyperparameter Tuning):

```
from sklearn.model_selection import
GridSearchCV
from sklearn.linear_model import
LogisticRegression
from sklearn.metrics import accuracy_score,
classification_report

param_grid = {
    'C': [0.001, 0.01, 0.1, 1, 10, 100],
    'penalty': ['l1', 'l2'],
    'solver': ['liblinear'],
    'class_weight': [None, 'balanced']
}

log_reg =
LogisticRegression(random_state=42)

grid_search = GridSearchCV(
    estimator=log_reg,
    param_grid=param_grid,
    scoring='f1',
    cv=5,
    n_jobs=-1
)

grid_search.fit(X_train, y_train)

best_model = grid_search.best_estimator_

print("Best Parameters:",
grid_search.best_params_)

y_pred_tuned = best_model.predict(X_test)
accuracy_tuned = accuracy_score(y_test,
y_pred_tuned)
print(f"Accuracy with best model:
{accuracy_tuned:.4f}")

class_report_tuned =
classification_report(y_test, y_pred_tuned)
print('\nClassification Report with best
model:')
print(class_report_tuned)
```

Output: (Hyperparameter Tuning):

```
Best Parameters: {'C': 1, 'class_weight': None, 'penalty': 'l2', 'solver': 'liblinear'}
Accuracy with best model: 0.9605

Classification Report with best model:
              precision    recall  f1-score   support

           0       0.97      0.99      0.98     18300
           1       0.86      0.64      0.73      1700

    accuracy                           0.96     20000
   macro avg       0.91      0.81      0.86     20000
weighted avg       0.96      0.96      0.96     20000
```

Conclusion:

- In this experiment, Linear Regression and Logistic Regression were studied and implemented on real-world datasets. Linear Regression was applied to the Salary–Experience dataset to predict salary based on work experience, demonstrating its effectiveness in modeling continuous outcomes. Logistic Regression was implemented on the Diabetes Prediction dataset to classify individuals as diabetic or non-diabetic based on medical and lifestyle features. The theoretical concepts of both regression techniques were successfully validated through practical implementation, highlighting their suitability for regression and classification tasks in real-world scenarios.