Lab Assignment: 7

Objective: To implement Logistic Regression algorithm and apply on a dataset.

Name: Aakash Verma

Reg. No.: 24-08-26

Course: M.Tech.(Cyber Security)

In [40]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
```

In [41]:
```
# Load the Pima Indians Diabetes Dataset
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes
column_names = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
                'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome']
diabetes_data = pd.read_csv(url, header=None, names=column_names)
```

In [42]:
```
# Split the dataset into features and target variable
X = diabetes_data.drop('Outcome', axis=1).values
y = diabetes_data['Outcome'].values

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42
```

In [43]:
```
# Feature scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [31]:
```
# Convert to DataFrame for easier handling
iris_df = pd.DataFrame(data=np.c_[X, y], columns=iris.feature_names + ['label'])

# Convert labels to binary for simplicity (e.g., class 0 vs. class 1 and 2)
y_binary = (y == 0).astype(int)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y_binary, test_size=0.2, random_s
```

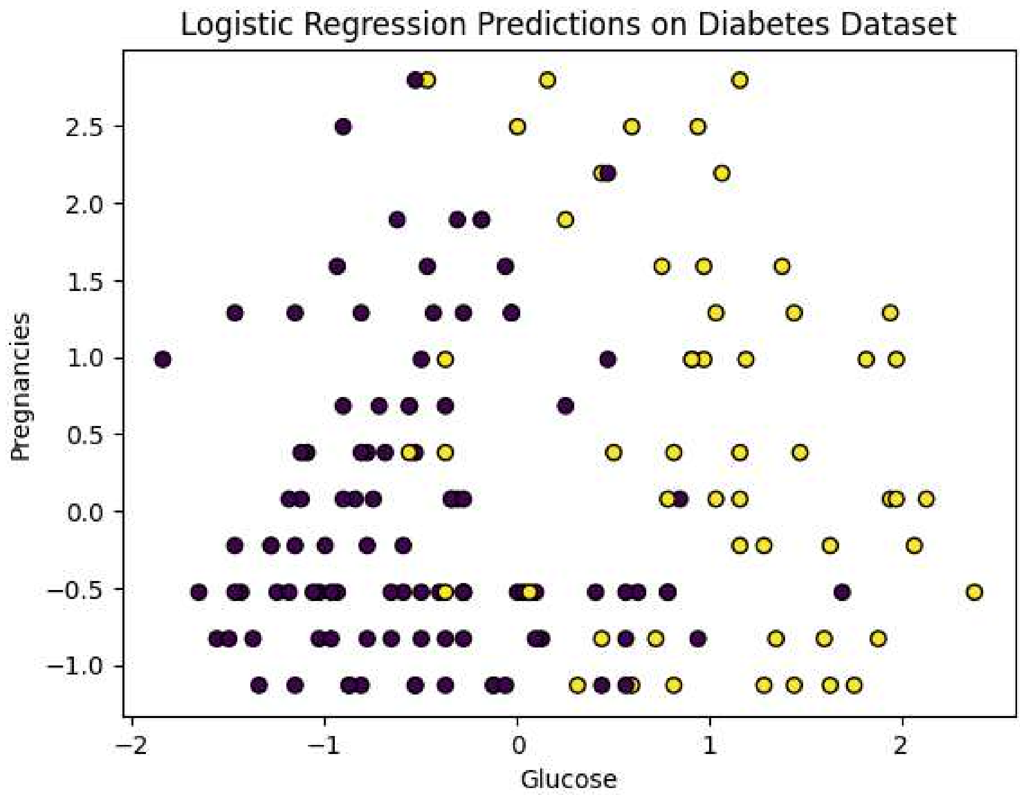
In [44]:
```
# Sigmoid function
def sigmoid(z):
    return 1 / (1 + np.exp(-z))

# Logistic Regression class
class LogisticRegression:
    def __init__(self, learning_rate=0.01, n_iterations=1000):
        self.learning_rate = learning_rate
        self.n_iterations = n_iterations
        self.weights = None
        self.bias = None

    def fit(self, X, y):
        n_samples, n_features = X.shape
        self.weights = np.zeros(n_features)
        self.bias = 0

        for _ in range(self.n_iterations):
            linear_model = np.dot(X, self.weights) + self.bias
            y_predicted = sigmoid(linear_model)

            # Gradient descent
            dw = (1 / n_samples) * np.dot(X.T, (y_predicted - y))
            db = (1 / n_samples) * np.sum(y_predicted - y)
            self.weights -= self.learning_rate * dw
            self.bias -= self.learning_rate * db

    def predict(self, X):
        linear_model = np.dot(X, self.weights) + self.bias
        y_predicted = sigmoid(linear_model)
        return [1 if i > 0.5 else 0 for i in y_predicted]
```

In [45]:
```
# Create and fit the Logistic Regression model
logistic_model = LogisticRegression(learning_rate=0.1, n_iterations=1000)
logistic_model.fit(X_train, y_train)
```

In [46]:
```
# Make predictions
predictions = logistic_model.predict(X_test)

# Calculate accuracy
accuracy = np.mean(predictions == y_test)
print(f"Accuracy from scratch: {accuracy:.2f}")
```

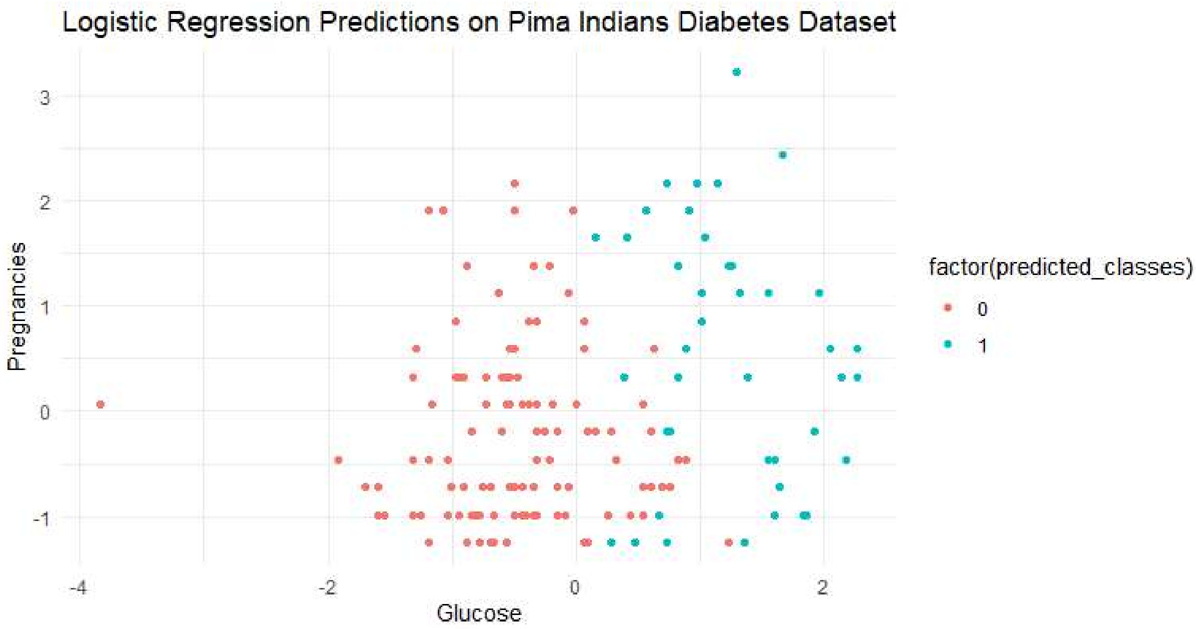
```
Accuracy from scratch: 0.75
```

In [47]:
```
# Visualization of results
plt.scatter(X_test[:, 1], X_test[:, 0], c=predictions, cmap='viridis', marker='o', edgec
plt.title("Logistic Regression Predictions on Diabetes Dataset")
plt.xlabel("Glucose")
plt.ylabel("Pregnancies")
plt.show()
```



R code

In [ ]:
```
# Install and Load necessary Libraries
install.packages("nnet")  # Uncomment if you haven't installed it
library(nnet)
library(ggplot2)

# Load the Pima Indians Diabetes Dataset
url <- "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabete
column_names <- c('Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness',
                  'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome')
diabetes_data <- read.csv(url, header = FALSE, col.names = column_names)

# Preview the dataset
head(diabetes_data)

# Convert Outcome to a factor
diabetes_data$Outcome <- as.factor(diabetes_data$Outcome)

# Split the dataset into features and target variable
set.seed(42)  # For reproducibility
train_indices <- sample(1:nrow(diabetes_data), size = 0.8 * nrow(diabetes_data))
train_data <- diabetes_data[train_indices, ]
test_data <- diabetes_data[-train_indices, ]

# Feature scaling
train_data[-ncol(train_data)] <- scale(train_data[-ncol(train_data)])
test_data[-ncol(test_data)] <- scale(test_data[-ncol(test_data)])

# Create and fit the Logistic Regression model
logistic_model <- glm(Outcome ~ ., data = train_data, family = binomial)

# Summary of the model
summary(logistic_model)

# Make predictions on the test data
predicted_probs <- predict(logistic_model, newdata = test_data, type = "response")
predicted_classes <- ifelse(predicted_probs > 0.5, 1, 0)  # Thresholding

# Calculate accuracy
accuracy <- mean(predicted_classes == as.numeric(as.character(test_data$Outcome)))
cat("Accuracy:", round(accuracy * 100, 2), "%\n")

# Confusion matrix
confusion_matrix <- table(Predicted = predicted_classes, Actual = as.numeric(as.characte
print("Confusion Matrix:")
print(confusion_matrix)

# Visualization of results
ggplot(test_data, aes(x = Glucose, y = Pregnancies, color = factor(predicted_classes))) +
  geom_point() +
  labs(title = "Logistic Regression Predictions on Pima Indians Diabetes Dataset",
       x = "Glucose", y = "Pregnancies") +
  theme_minimal()
```



Conclusion:

Effective Prediction: Accurately identifies individuals at risk for diabetes, aiding targeted interventions.

Actionable Insights: Reveals key health metrics influencing diabetes risk, informing clinical decisions.

Efficient and Scalable: Handles large datasets quickly, making it suitable for real-world applications.

In [ ]: