Project: Predicting Diabetes Risk Based on Patient Data

Objective:

Create a predictive model to classify whether a patient is at risk of developing diabetes (binary classification: 0 = no, 1 = yes) based on multiple features such as age, BMI, blood pressure, glucose levels, and other physiological factors. The dataset contains multiple variables of varying scales and units, so rescaling is necessary. Additionally, regularization is used to improve model performance and prevent overfitting.

Dataset:

Dataset is available in the same folder.

Dataset Features:

- Pregnancies
- Glucose level
- Blood Pressure
- Skin Thickness
- Insulin level
- BMI (Body Mass Index)
- Diabetes Pedigree Function (a function that scores likelihood based on family history)
- Age
- Outcome (0 = non-diabetic, 1 = diabetic)

Steps to Complete the Project:

1. Data Preprocessing

- **Load the Dataset**: Import the dataset using Pandas or similar libraries.
- **Handle Missing Values**: Check for missing or zero values in the data (e.g., insulin and skin thickness might have zeros that should be handled). [You can code this, or you can also do this manually and clean up your dataset]
- **Rescaling**: Since the features like age, glucose levels, and BMI are on different scales, apply feature scaling techniques (e.g., Min-Max scaling or Standardization) to bring them onto the same scale. Use StandardScaler or MinMaxScaler from the sklearn.preprocessing module.

2. Train-Test Split

- Split the dataset into training and testing sets using train_test_split from sklearn.model_selection. [70% train and 30% test]

3. Implement Logistic Regression with Regularization

- **Logistic Regression Model**: Fit a logistic regression model using LogisticRegression from sklearn.linear_model.
- **Regularization**: Use **L2 regularization** (also called Ridge regression) or **L1 regularization** (Lasso) to avoid overfitting. L2 is used by default in logistic regression, but you can control the regularization strength using the C parameter (smaller values of C increase regularization).
    - **L1 Regularization**: LogisticRegression(penalty='l1', solver='liblinear')
    - **L2 Regularization**: LogisticRegression(penalty='l2')
- Experiment with different values for C (inverse of regularization strength) and see how regularization affects the model's performance.

4. Model Evaluation

- **Please plot cost function vs iteration.**
- **Calculate the percentage accuracy of your model. Feeding in your TEST dataset, what percentage of datapoints your model was able to correctly predict whether a patient is diabetic or not.**

Deliverables:

1. **Jupyter Notebook/Code**: A well-documented notebook or script that walks through the steps from loading and preprocessing the data to building, and evaluating logistic regression model.
2. **Report**: A brief report summarizing the following: -Write your report on the top of your NOTEBOOK in a MARKDOWN CELL.
    - The data preprocessing steps and any challenges encountered (e.g., missing values, rescaling).
    - The performance of the logistic regression model with and without regularization.
    - The impact of regularization on preventing overfitting.

Key Skills Practiced:

- Logistic Regression with Regularization (L1/L2)
- Rescaling of data (Standardization or Normalization)
- Model Evaluation (Precision)

Tools/Technologies:

- Python

- Pandas, NumPy for data manipulation
- Scikit-learn for machine learning models and evaluation
- Matplotlib/Seaborn for data visualization

This project will help you gain hands-on experience with **logistic regression**, **regularization** techniques, and **rescaling** while working on a real-world dataset.