**PART A**

**EXPERIMENT NO. 8**

**A.1 AIM: -** TO detect Breast Cancer Diagnosis via Logistic Regression

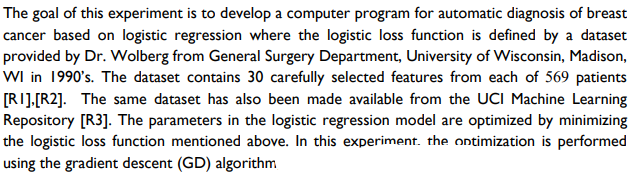
**A.2 Prerequisite**

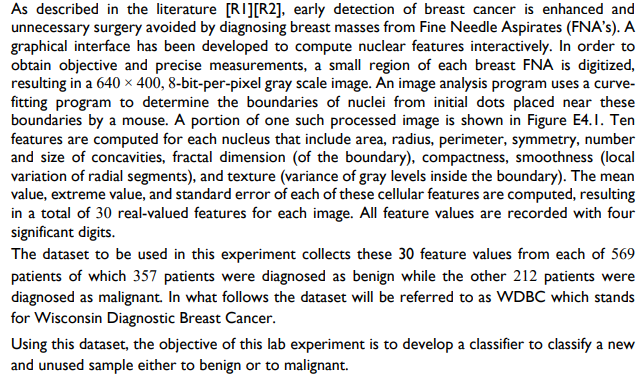
* Different programming language (Python or Java), Understanding of Machine Learning Algorithms, Machine Learning Algorithms

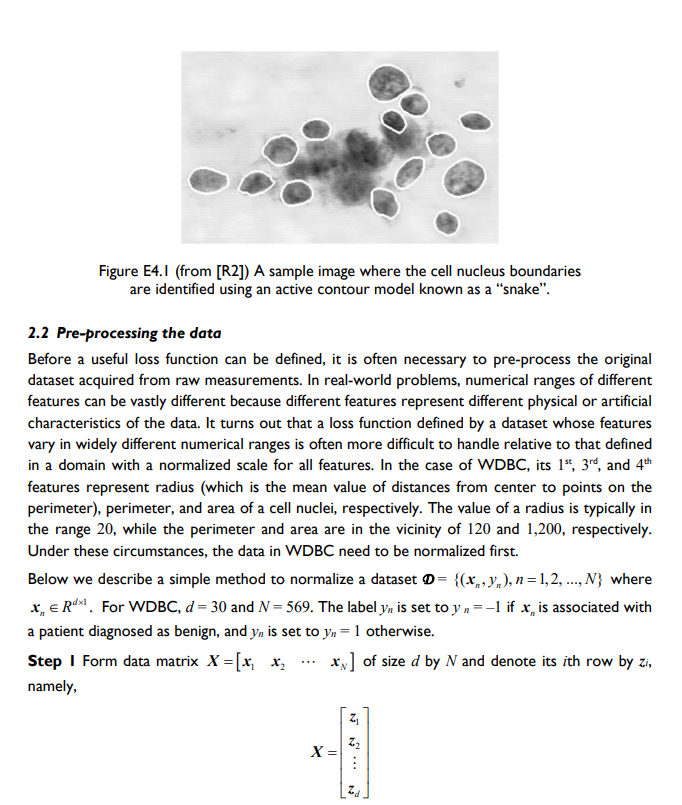
**A.3 Outcome**

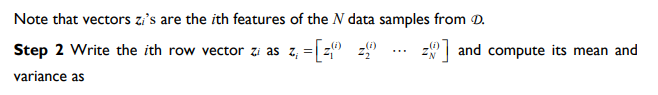
After successful completion of this experiment students will be able to Optimize the problem.

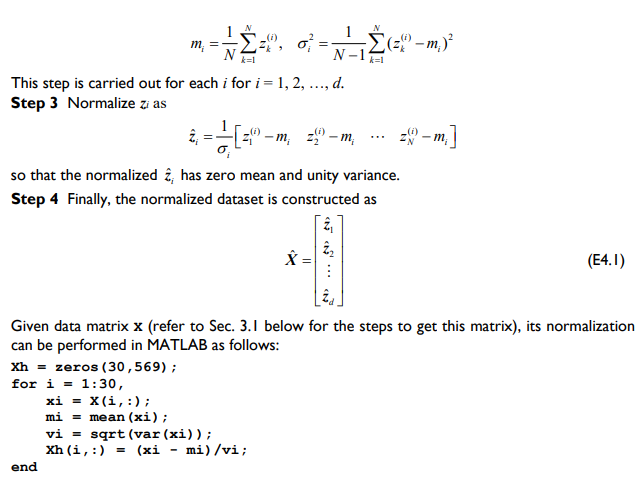
**A.4 Theory**



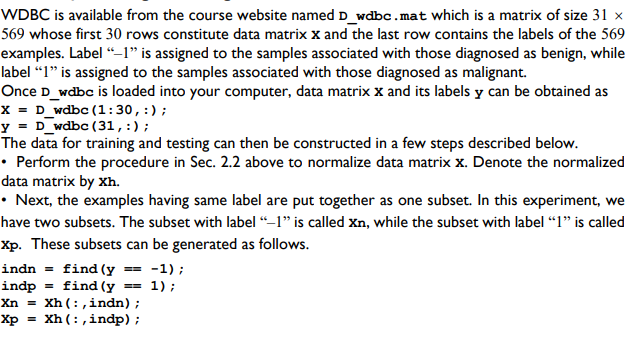


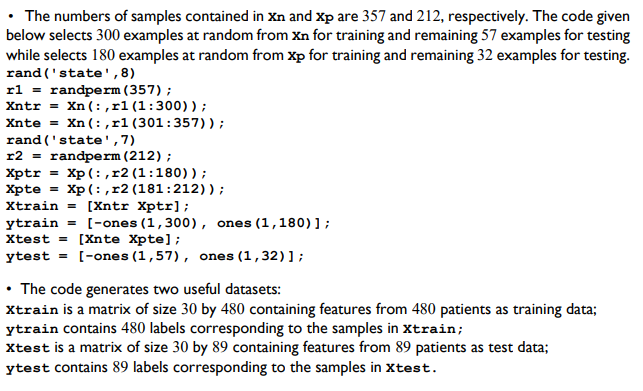


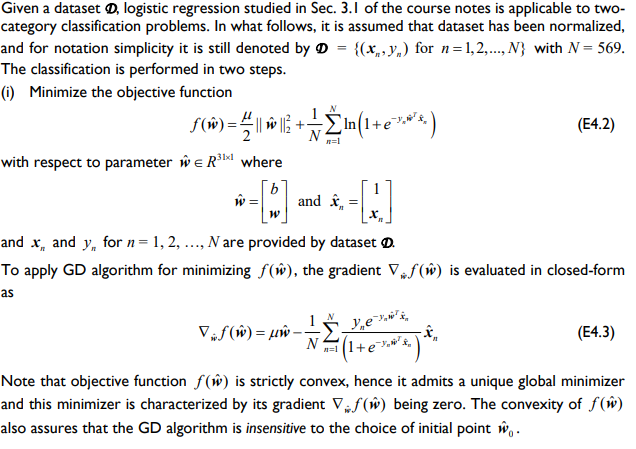


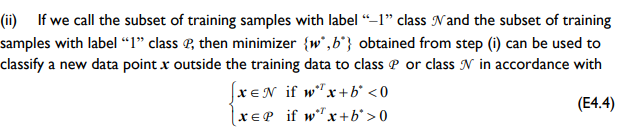


Procedure:









Links: <https://archive.ics.uci.edu/ml/datasets/breast%2Bcancer%2Bwisconsin%2B(Prognostic)>

Or

<https://www.kaggle.com/datasets/yasserhessein/dataset-breast-cancer-wisconsin>

PART B

|  |  |
| --- | --- |
| Roll No. C009 | Name: Samarth Borade |
| Class : BTI SEM 10 | Batch : EB1 |
| Date of Experiment: 22/03/24 | Date of Submission |
| Grade : |  |

**B.1 Documentation written by student:**

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split, GridSearchCV

from sklearn.linear\_model import LogisticRegression

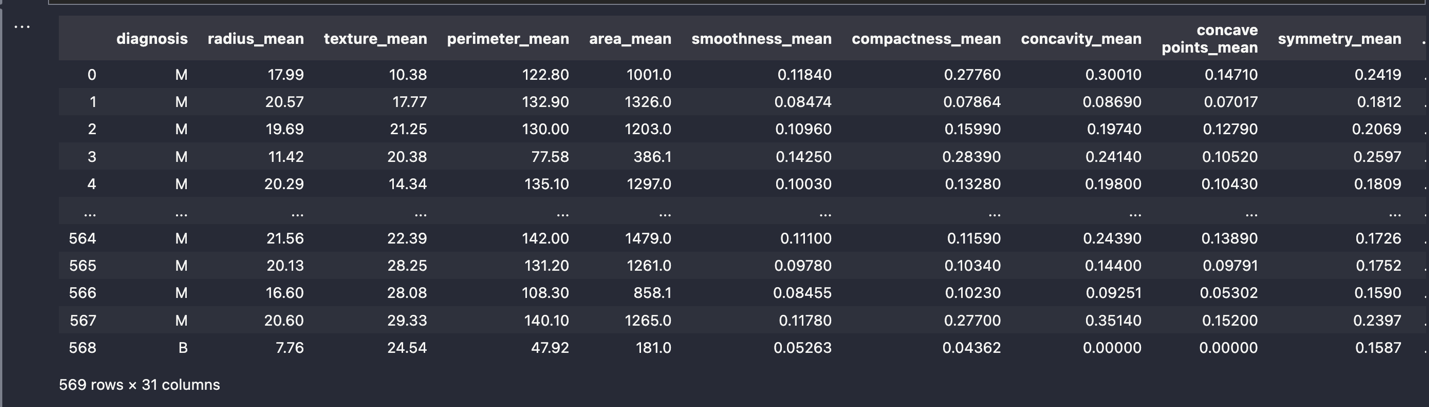
from sklearn.preprocessing import StandardScaler

from sklearn.metrics import classification\_report

data = pd.read\_csv('data.csv')

data.drop(["id", "Unnamed: 32"], *axis*=1,*inplace*=True)

data



from sklearn.metrics import accuracy\_score

X = data.iloc[:, 1:]

Y = data.iloc[:, 0]

# Convert target variable to binary (0, 1)

Y = np.where(Y == 'B', 0, 1)

scaler = StandardScaler()

X = scaler.fit\_transform(X)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, *random\_state*=42, *test\_size*=0.2)

log\_reg = LogisticRegression()

log\_reg.fit(X\_train, y\_train)

preds = log\_reg.predict(X\_test)

accuracy = accuracy\_score(y\_test, preds)

print("Accuracy of Logistic Regression (without optimization):", accuracy)

from sklearn.model\_selection import GridSearchCV

param\_grid = {

"penalty": ["l1", "l2"],

"C": [0.1, 1, 10],

"solver": ["liblinear", "saga"]

}

# Perform GridSearchCV

grid\_search = GridSearchCV(LogisticRegression(), *param\_grid*=param\_grid, *cv*=5)

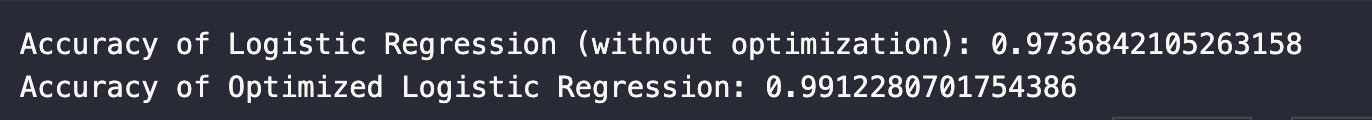
grid\_search.fit(X\_train, y\_train)

best\_log\_reg = grid\_search.best\_estimator\_

optimized\_preds = best\_log\_reg.predict(X\_test)

optimized\_accuracy = accuracy\_score(y\_test, optimized\_preds)

print("Accuracy of Optimized Logistic Regression:", optimized\_accuracy)



**B.2 Observations and learning:**

A graph with green and red lines

Description automatically generated

Upon comparing the performance of the logistic regression model before and after optimization, notable differences are evident. Initially, the unoptimized model exhibits a commendable accuracy of 97.37%. However, through the application of GridSearchCV for hyperparameter tuning, the optimized logistic regression model achieves a marked improvement, attaining an accuracy of 99.12%. This substantial enhancement underscores the significance of parameter optimization in refining model performance.

**B.3 Conclusion:**

In simple terms, tweaking the settings of a logistic regression model made a big difference. By fine-tuning these settings kind of like adjusting a recipe we boosted our accuracy to over 99%! This means we can trust our model more when making predictions. So, the lesson here is that making these adjustments is crucial for making sure our model works its best for sorting things into categories accurately.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*