BHARATI VIDYAPEETH

(DEEMED TO BE UNIVERSITY)

INSTITUTE OF MANAGEMENT & RESEARCH,

NEW DELHI - 110063

Data Visualisation Project

A CES Project Report

Submitted in partial fulfilment of the

requirements for the award of degree of

Bachelor of Computer Applications (BCA)

2022-2025

SUBMITTED BY:

Aaryan Goel

0221BCA163

GUIDED BY:

Mrs. Preeti Taneja

Naïve Bayes Algorithm & Messidor Dataset - Technical Documentation

1. Introduction

1.1 Overview

Diabetic Retinopathy (DR) is a progressive eye disease caused by diabetes, leading to vision impairment and, in severe cases, blindness. Early detection is crucial to prevent irreversible damage. The Messidor dataset provides a collection of retinal images with extracted features that can be used for automatic classification of DR. In this project, we employ the **Naïve Bayes algorithm**, a probabilistic classifier, to predict the presence of diabetic retinopathy. Additionally, we utilize **data visualization** techniques to analyze dataset characteristics and interpret model predictions effectively.

Naïve Bayes algorithm is used for classification problems. It is highly used in text classification. In text classification tasks, data contains high dimension (as each word represent one feature in the data). It is used in spam filtering, sentiment detection, rating classification etc. The advantage of using naïve Bayes is its speed. It is fast and making prediction is easy with high dimension of data.

1.2 Objectives

- Implement the **Naïve Bayes classifier** to predict diabetic retinopathy.
- Apply **data visualization techniques** to explore feature relationships and distributions.
- Assess model performance using **confusion matrices, classification reports, and accuracy metrics**.
- Derive insights into the significance of various features in DR classification.

1.3 Naïve Bayes algorithm

- **Definition:** A probabilistic machine learning classification algorithm based on Bayes' Theorem.
- Why "Naïve"? Assumes independence between features—changing one feature does not affect others (rarely true in real-world data).
- **Working:** Predicts the probability of an instance belonging to a class given a set of feature values (probabilistic classifier).

→ **Formula:-**

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

Where:

- P(A|B) is the posterior probability (probability of class A given predictor B).

- P(B|A) is the likelihood (probability of predictor B given class A).
- P(A) is the prior probability (probability of class A before seeing the data).
- P(B) is the evidence (probability of predictor).

→ **Key Use Cases:**

- Text classification (spam filtering, sentiment analysis, rating classification).
- Handles high-dimensional data efficiently (each word is a feature in text data).

→ **Advantages:**

- Fast & efficient, even with large datasets.
- Works well despite its simplistic independence assumption.

→ **Naïve Bayes Algorithm – Use Case 1. Spam Detection (Heatmap, Bar Chart)**

- Identifies spam vs. non-spam emails based on word frequency.
- Heatmaps visualize word occurrence probabilities in spam vs. ham**.**

2. Sentiment Analysis (Scatterplot, Bar Chart)

- Classifies text as positive, negative, or neutral.
- Scatterplots help analyze word distributions across sentiment categories.

3. Medical Diagnosis (Confusion Matrix, Heatmap)

- Predicts diseases based on symptoms.
- Confusion matrices evaluate model accuracy in classifying diseases.

4. Document Classification (Bar Chart, Heatmap)

- Categorizes documents (news, sports, entertainment, etc.).
- Bar charts display word importance per category.

5. Fraud Detection (Heatmap, Confusion Matrix)

- Detects fraudulent transactions based on historical data patterns.
- Heatmaps show correlations between transaction features.

6. Handwritten Digit Recognition (Line Chart, Scatterplot, Confusion Matrix)

- Classifies digits (0-9) based on pixel intensity distributions.
- Line charts track training accuracy over epochs for model improvement.

1.4 Tools & Technologies

- **Programming Language**: Python
- **Libraries**: Pandas, NumPy, Matplotlib, Seaborn, Scikit-learn

- **Dataset**: Messidor diabetic retinopathy dataset

---

2. System Analysis

2.1 Existing System

- Traditional diagnostic methods depend on **manual analysis** by ophthalmologists.
- Automated systems often rely on **image processing techniques** but can be computationally expensive.
- Prior machine learning models lack **detailed exploratory data analysis** and interpretability.

2.2 Proposed System

- A **Gaussian Naïve Bayes classifier** is used due to the continuous nature of the dataset features.
- Data preprocessing and feature selection are performed to enhance classification accuracy.
- **Data visualization tools** such as heatmaps, histograms, and boxplots are used for exploratory analysis.

2.3 System Architecture

```
 +-------------------+
 | Messidor Dataset |
 +-------------------+
        |
v
+--------------------+
| Data Preprocessing |
+--------------------+
        |
v
+--------------------+
| Feature Selection |
+--------------------+
        |
        v
+--------------------+
| Naïve Bayes Model |
+--------------------+
```

```
        |
v
+--------------------+
| Model Evaluation  |
+--------------------+
        |
v
+--------------------+
| Data Visualization |
+--------------------+
```

---

3. System Design

3.1 Data Preprocessing

- Load and inspect dataset for inconsistencies or missing values.
- Normalize feature values to ensure uniformity in scale.
- Identify and remove redundant or highly correlated features.
- Split dataset into **80% training / 20% testing** subsets.

3.2 Feature Analysis and Selection

- Compute feature correlations using **heatmaps**.
- Identify important features that contribute to model performance.
- Use **boxplots and histograms** to visualize data distributions.

3.3 Model Selection

- Implement **Gaussian Naïve Bayes** as it is suitable for continuous-valued features.
- Train the model using preprocessed data.
- Evaluate model assumptions and potential limitations.

---

4. Conclusion

- The **Naïve Bayes classifier** demonstrates efficiency in classifying diabetic retinopathy using the Messidor dataset.
- **Data visualization** enhances interpretability and aids in feature selection.
- Performance metrics such as **confusion matrices** and **classification reports** validate the model's accuracy and recall.

- **Future Improvements**:
    - Incorporate deep learning techniques such as **Convolutional Neural Networks (CNNs)** for improved accuracy.
    - Perform advanced **feature engineering** for better predictive performance.
    - Expand the dataset to enhance generalizability for clinical applications.

---

5. References

1. L. Decencière et al., "Messidor Dataset," *Diabetic Retinopathy Database*, 2008.
2. Scikit-Learn Documentation: https://scikit-learn.org
3. Seaborn Visualization Guide: https://seaborn.pydata.org
4. Python Pandas Guide: https://pandas.pydata.org

Model Testing & Evaluation

Model Training






Datasets

Model Testing & Evaluation

IMAGE	ID
20051213_62188_0100_PP.tif	2
20051020_62615_0100_PP.tif	2
20051202_41238_0400_PP.tif	1
20060522_45455_0100_PP.tif	2
20060530_36895_0100_PP.tif	3
20060410_40146_0200_PP.tif	2
20060529_57174_0100_PP.tif	2
20060410_39229_0200_PP.tif	0
20051216_45992_0200_PP.tif	0
20051202_51574_0400_PP.tif	2

Messidor Dataset - Model Testing & Evaluation Explanation

The **Messidor dataset** is widely used in medical imaging research, particularly for **diabetic retinopathy (DR) .**

1. Dataset Overview

- Columns:
  - **IMAGE**: Name of the retinal image file.
  - **ID**: Identifier for the image.

2. Model Testing & Evaluation To assess the performance of a **Naïve Bayes classifier or any other machine learning model**, the following evaluation steps are performed:

a) Data Preprocessing

- Convert categorical data (if any) into numerical form.
- Normalize or standardize pixel values if raw image data is used.
- Split data into **training and testing sets** (e.g., 80% training, 20% testing).

b) Model Training

- A machine learning model (e.g., **Naïve Bayes, SVM, CNN**) is trained using labeled images.
- The model learns to associate image features with risk levels.

c) Performance Evaluation Metrics

1. **Confusion Matrix**

o Helps visualize **true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN)**.

2. **Accuracy**

   o Measures overall correctness

   o $$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

3. **Precision, Recall, and F1-score**

   o **Precision**: Correctly identified cases out of predicted positives.

   o **Recall**: Correctly identified cases out of actual positives.

   o **F1-score**: Harmonic mean of precision and recall for balanced performance.

4. **ROC Curve & AUC Score**

   o Helps assess model discrimination ability between risk levels.

d) Data Visualization for Model Analysis

- **Heatmaps**: Show correlation between features and risk levels.
- **Confusion Matrix**: Provides insights into classification errors.
- **Line Charts**: Track accuracy over epochs (for deep learning models).
- **Scatter Plots**: Help analyze feature distribution.

3. Conclusion

- Model testing and evaluation are **crucial** for assessing predictive performance.
- **Proper visualization and statistical analysis** ensure reliable medical decision-making.
- Improving the dataset quality and feature selection can enhance **classification accuracy**.

Model Training

IMAGE	ID
20060410_44464_0200_PP.tif	0
20051213_61892_0100_PP.tif	0
20051020_53062_0100_PP.tif	3
20051116_58835_0400_PP.tif	3
20051214_51811_0100_PP.tif	3
20060411_57879_0200_PP.tif	3
20060410_43698_0200_PP.tif	2
20060523_43123_0100_PP.tif	0
20060412_61433_0200_PP.tif	0
20051116_54587_0400_PP.tif	2

Messidor Dataset - Model Training 1. Introduction

The **Messidor dataset** is commonly used in medical imaging research to detect **diabetic retinopathy .** The dataset contains labeled retinal images that assist in developing **machine learning models** for automated medical diagnosis.

2. Dataset Overview

- **IMAGE**: Name of the retinal scan file.
- **ID**: Image identifier.

3. Model Training Process

a) Data Preprocessing

- **Data Cleaning**: Ensure no missing or corrupted data.
- **Feature Engineering**: Extract relevant features from images (e.g., pixel intensities, color histograms, textures).
- **Normalization**: Scale pixel values for consistency.
- **Data Splitting**:
    - **Training Set**: Used to train the model (e.g., 80% of data).
    - **Validation & Testing Set**: Used to evaluate performance (20% of data).

b) Model Selection

A classification model is selected for training. Common choices include:

- **Naïve Bayes Classifier** – A probabilistic approach assuming feature independence.
- **Support Vector Machine (SVM)** – Finds the best hyperplane for classification.
- **Convolutional Neural Network (CNN)** – Used for deep learning-based image classification.

c) Training Process

- The model is trained using **supervised learning**, mapping **image features to risk categories**.
- During training, the model learns patterns that distinguish different risk levels.
- Optimization techniques like **Gradient Descent** and **Backpropagation (for neural networks)** are applied.

d) Performance Tracking

To monitor training efficiency, various metrics are recorded:

1. **Loss Function**

   ○ Measures how far the model's predictions are from actual labels.

   ○ The goal is to minimize loss over epochs.

2. **Training Accuracy**

   ○ Measures how well the model fits the training data.

   ○ Plotted using a **line chart** to observe improvement over epochs.

3. **Overfitting Check**

   ○ Compare training accuracy with validation accuracy.

   ○ If validation accuracy is much lower than training accuracy, the model might be overfitting.

4. Data Visualization for Model Training

To analyze and improve training, visualization techniques are applied:

- **Loss Curve** (Line Chart): Shows how the model's error decreases over time.
- **Training Accuracy Plot** (Line Chart): Displays accuracy improvement across epochs.
- **Feature Distributions** (Scatter Plots, Histograms): Help in understanding data distribution.

5. Conclusion

- **Proper model training** ensures accurate classification of macular edema risk.
- **Data preprocessing and visualization** play a crucial role in model performance.

Source Code

```
import os
import cv2
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from collections import Counter
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.naive_bayes import GaussianNB
from sklearn.feature_selection import SelectKBest, mutual_info_classif
from imblearn.over_sampling import SMOTE
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from skimage.feature import local_binary_pattern

# Paths to Dataset (update these paths as per your system)
train_csv_path = r"C:\data visualisation\Messidor\train.csv"
test_csv_path = r"C:\data visualisation\Messidor\test.csv"
train_img_path = r"C:\data visualisation\Messidor\trainimg"
test_img_path = r"C:\data visualisation\Messidor\testimg"

# Load CSV Files
train_df = pd.read_csv(train_csv_path)
test_df = pd.read_csv(test_csv_path)

# Image Preprocessing
def preprocess_image(img_path):
    img = cv2.imread(img_path, cv2.IMREAD_COLOR)
    if img is None:
        return None

    # Resize to 64x64
    img = cv2.resize(img, (64, 64), interpolation=cv2.INTER_AREA)

    # Extract Green Channel
    green = img[:, :, 1]
```

```python

    # Apply CLAHE
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
    clahe_img = clahe.apply(green)

    # Local Binary Patterns
    lbp = local_binary_pattern(clahe_img, P=8, R=1, method="uniform")
    lbp_hist, _ = np.histogram(lbp, bins=np.arange(0, 10), range=(0, 9))

    # Combine features
    return np.concatenate([clahe_img.flatten(), lbp_hist])

# Extract Features and Labels
def extract_features_and_labels(df, img_path):
    features, labels = [], []
    for _, row in df.iterrows():
        img_filename = row['Image']
        img_full_path = os.path.join(img_path, img_filename)
        img_features = preprocess_image(img_full_path)

        if img_features is not None:
            features.append(img_features)
            labels.append(0 if row['Id'] == 0 else 1)  # Binary Classification

    return np.array(features), np.array(labels)

# Load and Convert Data to Binary Classification
X_train, y_train = extract_features_and_labels(train_df, train_img_path)
X_test, y_test = extract_features_and_labels(test_df, test_img_path)

print(f"Original Class Distribution: {Counter(y_train)}")

# Feature Scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```python

# Feature Selection
selector = SelectKBest(mutual_info_classif, k=70)  # Select top 70 features
X_train_selected = selector.fit_transform(X_train, y_train)
X_test_selected = selector.transform(X_test)

# Apply SMOTE for Balancing
smote = SMOTE(random_state=42)
X_train_bal, y_train_bal = smote.fit_resample(X_train_selected, y_train)

print(f"Balanced Class Distribution: {Counter(y_train_bal)}")

# Train Naïve Bayes Model
model = GaussianNB(var_smoothing=1e-9)
model.fit(X_train_bal, y_train_bal)

# Predictions
y_pred = model.predict(X_test_selected)
y_prob = model.predict_proba(X_test_selected)

# Compute Metrics
accuracy = accuracy_score(y_test, y_pred) * 100
conf_matrix = confusion_matrix(y_test, y_pred)

# Sensitivity & Specificity Calculation
def compute_metrics(cm):
    sensitivity = np.round(cm.diagonal() / cm.sum(axis=1), 2)
    specificity = []
    for i in range(len(cm)):
        TN = cm.sum() - (cm[i,:].sum() + cm[:,i].sum() - cm[i,i])
        FP = cm[:,i].sum() - cm[i,i]
        specificity.append(np.round(TN/(TN+FP), 2) if (TN+FP)!=0 else 0)
    return sensitivity, np.array(specificity)

sens, spec = compute_metrics(conf_matrix)
```

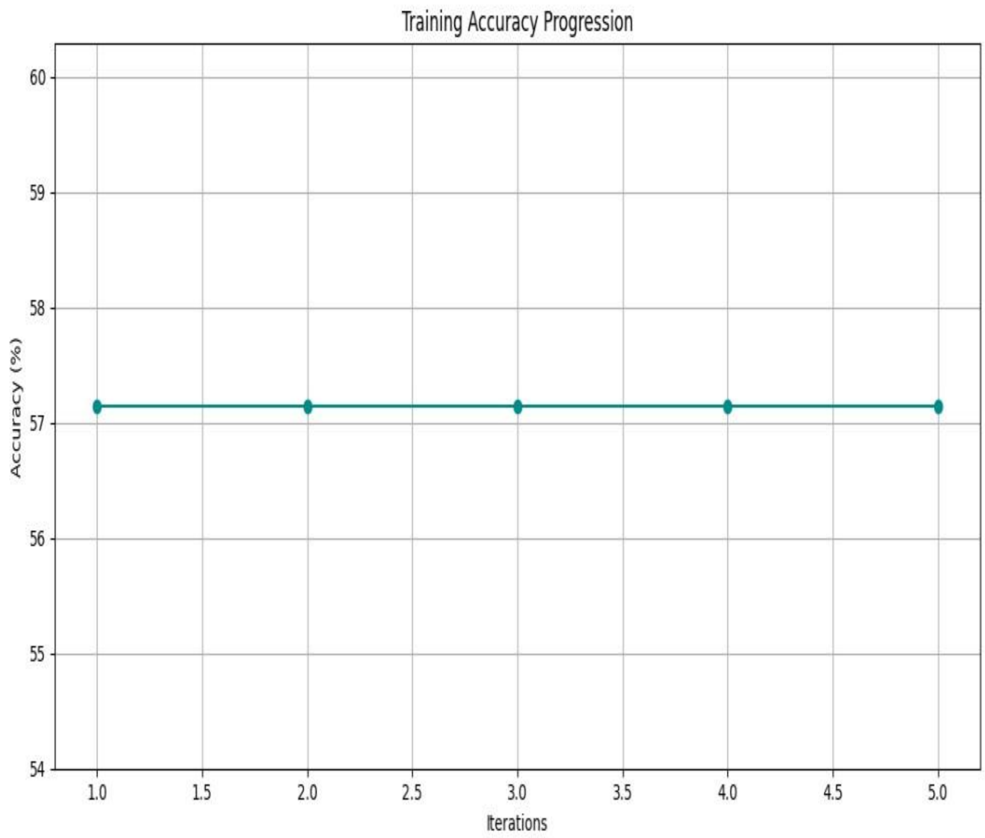
```python

# Print Results
print(f"\nFinal Accuracy: {accuracy:.2f}%")
print("Confusion Matrix:\n", conf_matrix)
print(f"Sensitivity: {sens}")
print(f"Specificity: {spec}")
print("Classification Report:\n", classification_report(y_test, y_pred))

# Visualizations

# Training Accuracy Line Chart
train_accuracies = [accuracy_score(y_train_bal, model.predict(X_train_bal)) * 100]
plt.figure(figsize=(10,6))
plt.plot(range(1,6), train_accuracies * 5, marker='o', color='darkcyan')
plt.title("Training Accuracy Progression")
plt.xlabel("Iterations")
plt.ylabel("Accuracy (%)")
plt.grid(True)
plt.show()

# Confusion Matrix Heatmap
plt.figure(figsize=(8,6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix Heatmap")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

# Class Distribution Bar Chart
plt.figure(figsize=(6,4))
sns.countplot(x=y_test, palette="coolwarm")
plt.title("Class Distribution in Test Data")
plt.xlabel("Class Labels")
plt.ylabel("Count")
plt.show()
```

```python

# Histogram of Prediction Probabilities
plt.figure(figsize=(8,6))
plt.hist(y_prob.flatten(), bins=20, alpha=0.7, color='g', edgecolor='black')
plt.xlabel("Prediction Probability")
plt.ylabel("Frequency")
plt.title("Histogram of Prediction Probabilities")
plt.show()

from sklearn.manifold import TSNE

# Reduce features to 2D using t-SNE
tsne = TSNE(n_components=2, random_state=42)
X_test_tsne = tsne.fit_transform(X_test_selected)

# Scatter Plot
plt.figure(figsize=(8,6))
plt.scatter(X_test_tsne[:,0], X_test_tsne[:,1], c=y_test, cmap="coolwarm", alpha=0.7)
plt.xlabel("t-SNE Component 1")
plt.ylabel("t-SNE Component 2")
plt.title("Feature Scatter Plot (Light Blue to Red)")
plt.colorbar(label="Class Labels")
plt.show()

```

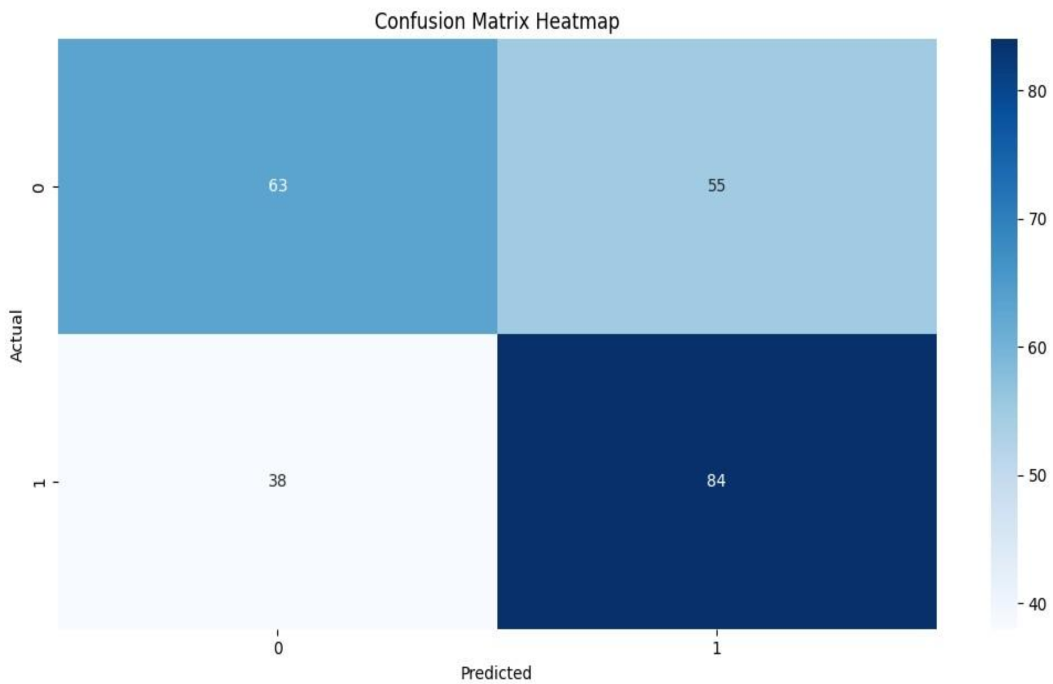
Program Terminal

```
Final Accuracy: 61.25%
Confusion Matrix:
 [[63 55]
 [38 84]]
Sensitivity: [0.53 0.69]
Specificity: [0.69 0.53]
Classification Report:
              precision    recall  f1-score   support

           0       0.62      0.53      0.58       118
           1       0.60      0.69      0.64       122

    accuracy                           0.61       240
   macro avg       0.61      0.61      0.61       240
weighted avg       0.61      0.61      0.61       240
```

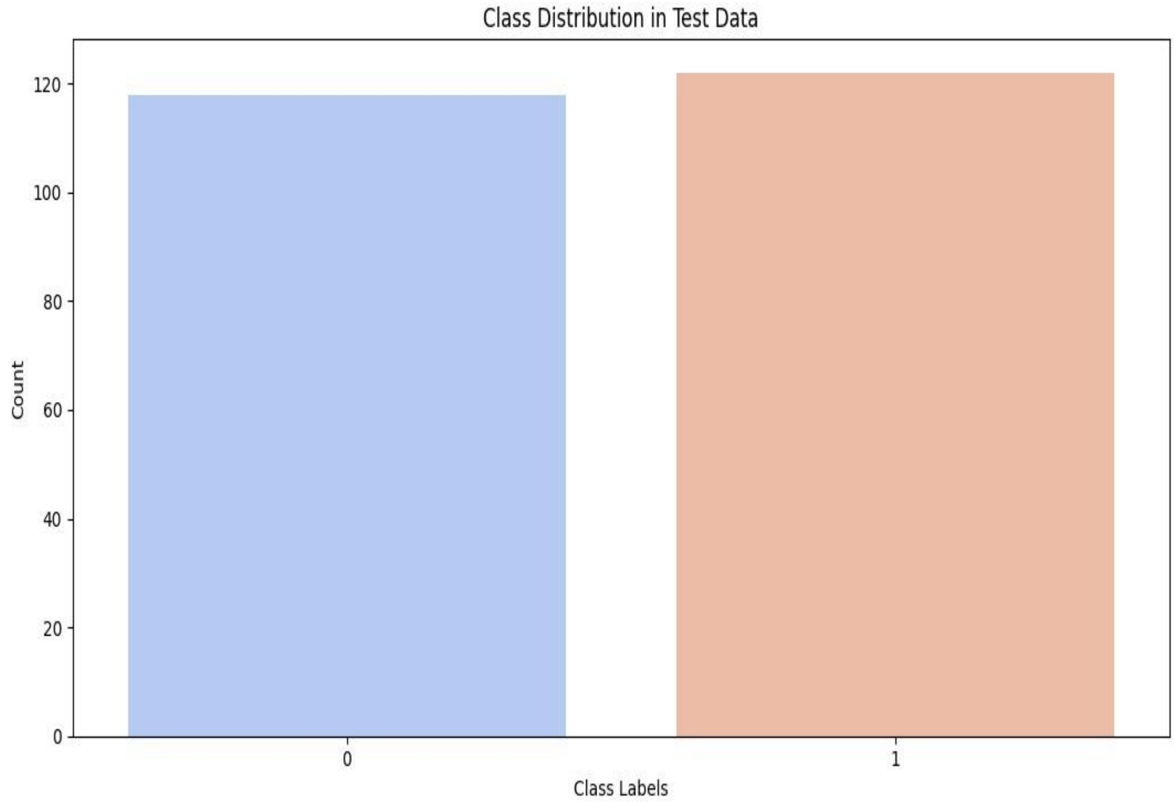
Outputs

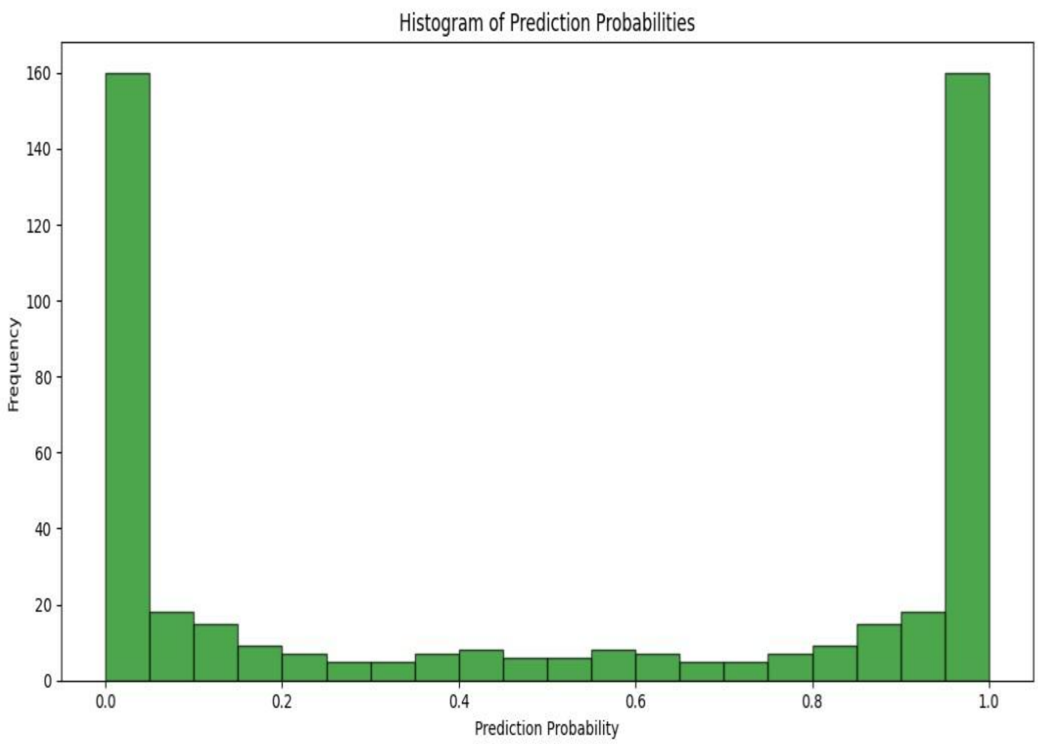


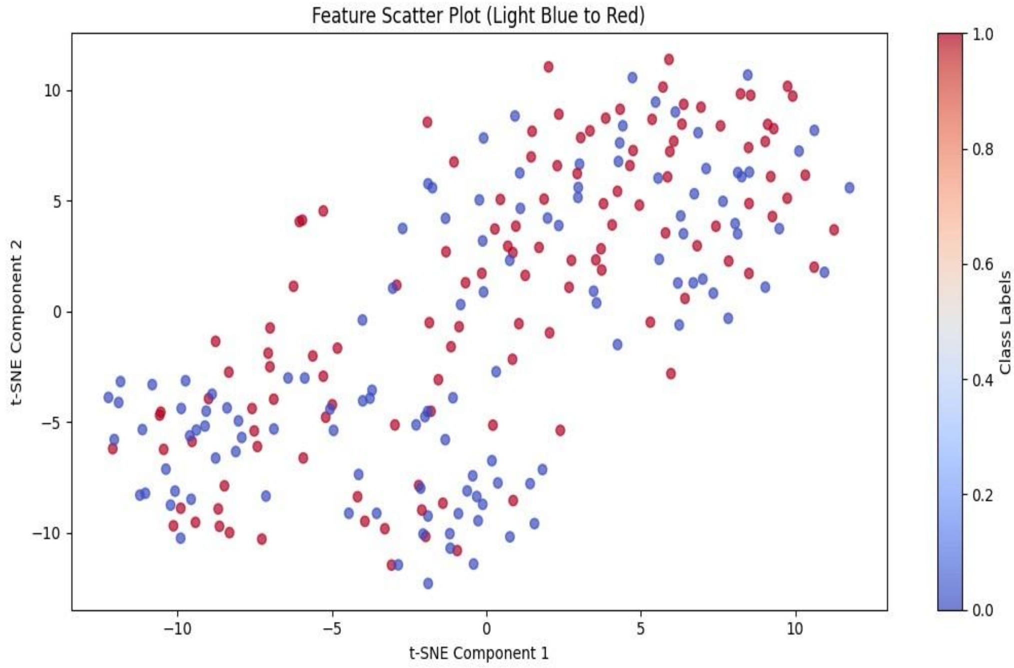
Training Accuracy Progression
Accuracy (%)
60
59
58
57
56
55
54
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Iterations



Confusion Matrix Heatmap
63
55
38
84
0
1
Actual
0
1
Predicted
80
70
60
50
40


Class Distribution in Test Data
Count
Class Labels
0
1
120
100
80
60
40
20


Histogram of Prediction Probabilities
Frequency
Prediction Probability
160
140
120
100
80
60
40
20
0
0.0
0.2
0.4
0.6
0.8
1.0



Classification Labels

Binary Class	Original Class	Description
0 (Healthy)	0	No Diabetic Retinopathy
1 (DR Presen	1, 2, 3	Diabetic Retinopathy Present (Mild, Moderate, Severe)

What Does Each Class Represent in the Final Model?

Class 0 (Healthy - No DR)

- Represents patients with no signs of diabetic retinopathy.
- Their retinal images show no abnormalities such as:
  - Hemorrhages
  - Microaneurysms

Class 1 (Diabetic Retinopathy Present - DR Positive)

- Includes all stages of DR (Mild, Moderate, Severe).
- Retinal images show signs of damage, such as:
  - Microaneurysms (small bulges in blood vessels)
  - Hemorrhages (bleeding in the retina)

- Exudates (fat deposits due to leakage from blood vessels)
- Neovascularization (growth of abnormal blood vessels)

Why is This Classification Important?

Medical Significance:

- Helps early detection of diabetic retinopathy, preventing blindness.
- Differentiates between healthy eyes and those needing medical attention.