Image Classification Model for Diabetic_Retinopathy Documentation

1. Dataset Overview

The dataset used for this image classification task is from the [Kaggle Diabetic Retinopathy 224x224 2019 Dataset].

1.1 Dataset Type

- **Type:** Image dataset (RGB colored images)
- **Image Dimensions:** 224x224 pixels

1.2 Classes

The dataset consists of 5 distinct classes that represent the stages of diabetic retinopathy:

- **No_DR:** No diabetic retinopathy detected.
- **Mild:** Mild diabetic retinopathy.
- **Moderate:** Moderate diabetic retinopathy.
- **Severe:** Severe diabetic retinopathy.
- **Proliferate_DR:** Proliferative diabetic retinopathy.

1.3 Dataset Distribution

- **Total Images:** 3662 images
- **Train-Test Split:** The dataset was split into 80% for training and 20% for testing.
- **Class Imbalance:** The classes "Severe" and "Proliferate_DR" contain fewer images (192 and

295 images, respectively), which could potentially affect model performance.

1.4 Missing Data

There is no missing data in the dataset. However, the class imbalance could be a challenge during training, as some classes have very few images for model training.

---

2. Data Preprocessing

The preprocessing of the dataset involved several important steps to prepare it for training:

2.1 Feature Extraction

The dataset comes with a CSV file that includes two columns: `diagnosis` (labels) and `id_code` (image IDs). These columns were used to map numeric diagnosis values to the corresponding class labels:

- 0 → No_DR
- 1 → Mild
- 2 → Moderate
- 3 → Severe
- 4 → Proliferate_DR

2.2 Image Processing

- **Resizing**: All images were resized to 64x64 pixels to ensure uniformity across the dataset.

- **Flattening**: The resized images were converted into one-dimensional feature vectors, which are easier for machine learning models to process.
- **Normalization**: Feature vectors were standardized using a scaling technique to ensure that the input features are centered around zero with a unit variance.

2.3 Train-Test Split

The data was split into training and testing sets using an 80%-20% split. The stratification process ensured that the class distribution in both the training and testing sets remained proportional to the original dataset.

---

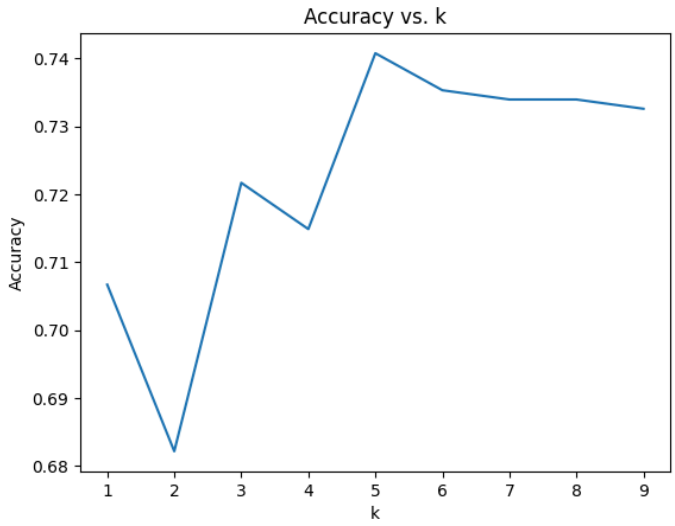
3. Models Used

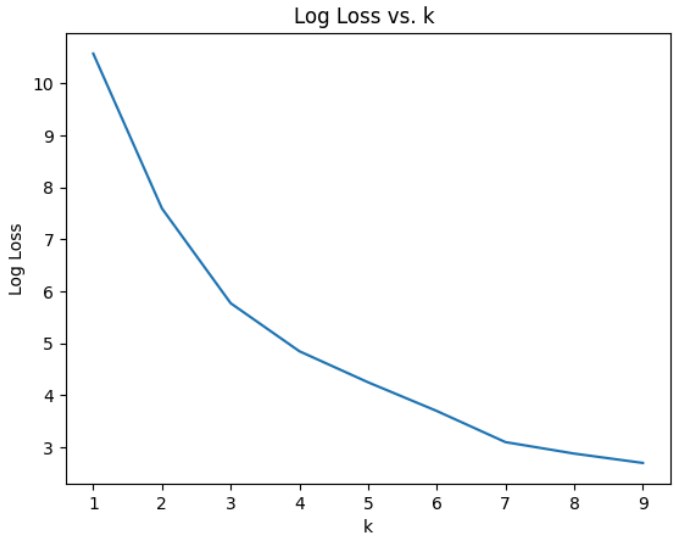
3.1 K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a simple yet powerful classification algorithm. We experimented with different values of `k`, the number of neighbors, to determine the optimal value for the model.

We trained the KNN model with values of `k` ranging from 1 to 9 and plotted the accuracy and log loss for each value.

- **Best Accuracy**: The best accuracy was achieved when `k=5`, with a performance of **0.7408**.

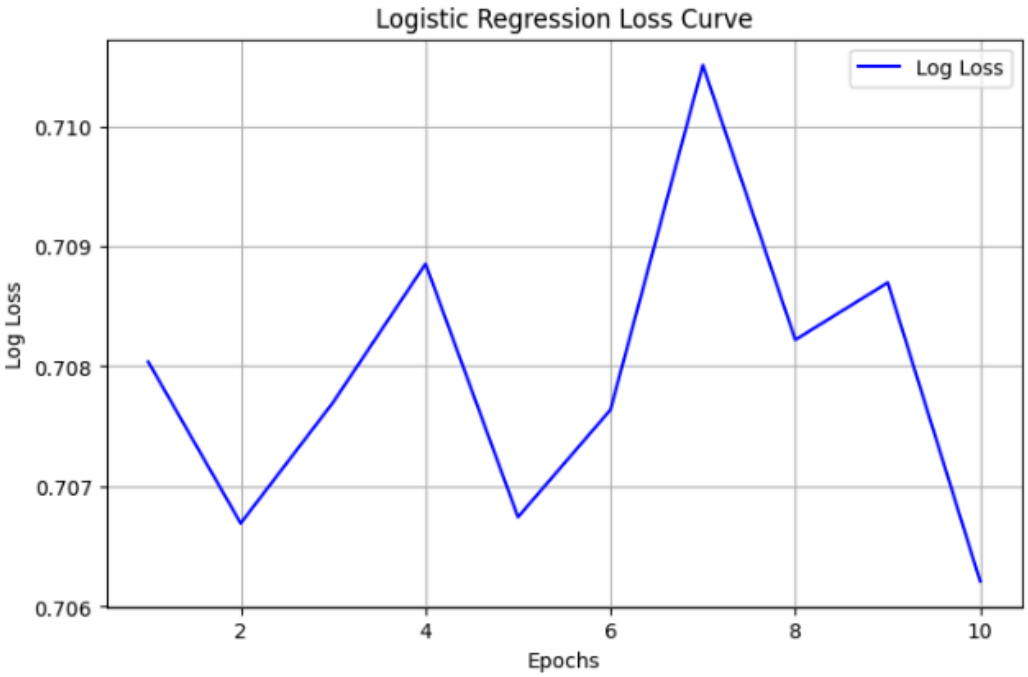




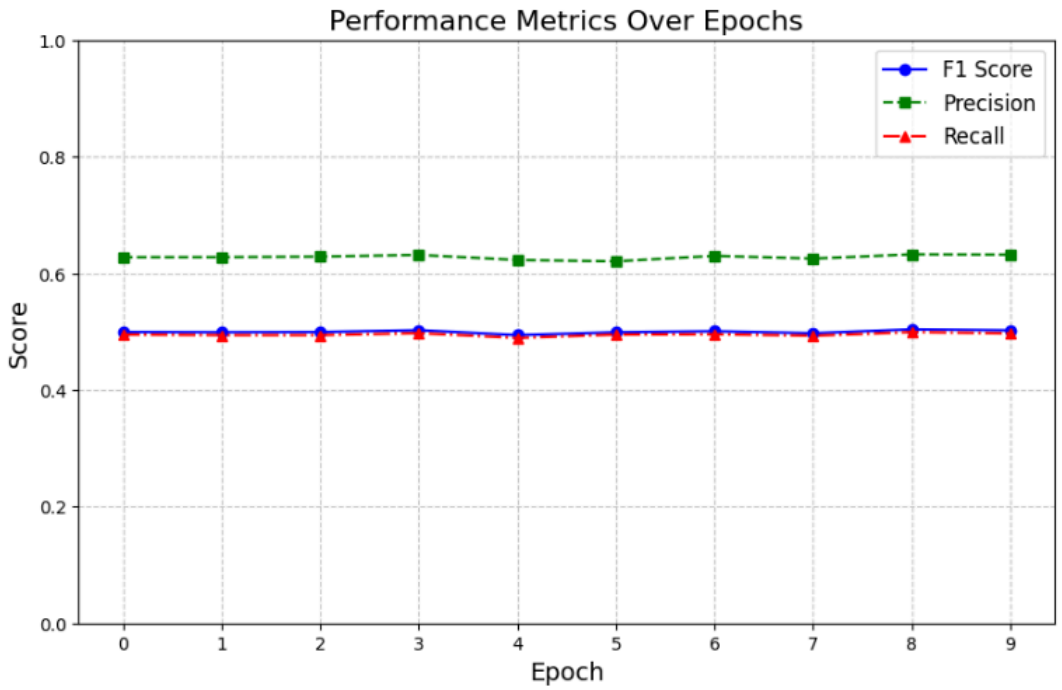
3.2 Logistic Regression (LR)

Logistic Regression (LR) was trained iteratively over 10 epochs. This allowed us to observe how the model improved in terms of various evaluation metrics as it was trained over multiple epochs.

- **Final Accuracy:** After training for 10 epochs, the Logistic Regression model achieved an accuracy of **0.7531**



We tracked several performance metrics, including precision, recall, and F1 score, for each epoch. These metrics were compared to assess the model's performance over time.



3.3 Decision Tree Classifier (“Extra”)

The Decision Tree Classifier was also evaluated, providing another point of comparison for our models. While it performed decently, it did not outperform the KNN or LR models.

- **Accuracy**: The Decision Tree model achieved an accuracy of **0.6726.**

3.4 SVM (“Extra”)

The SVM was also implemented, as another point of difference for our classification model.it didn’t quite reach the level we were expecting.

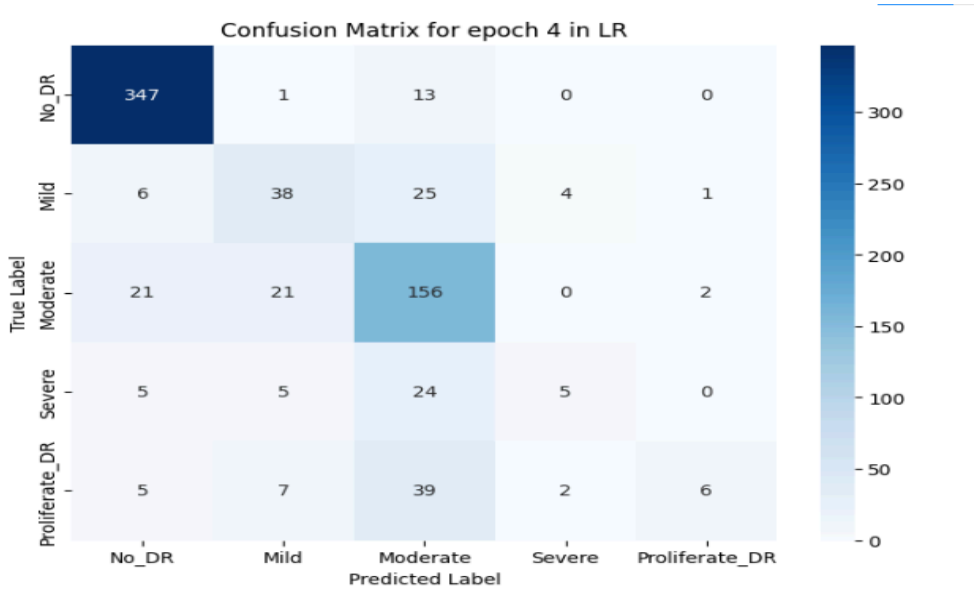
Accuracy: The Decision Tree model achieved an accuracy of **0.5771.**

---

4. Model Evaluation

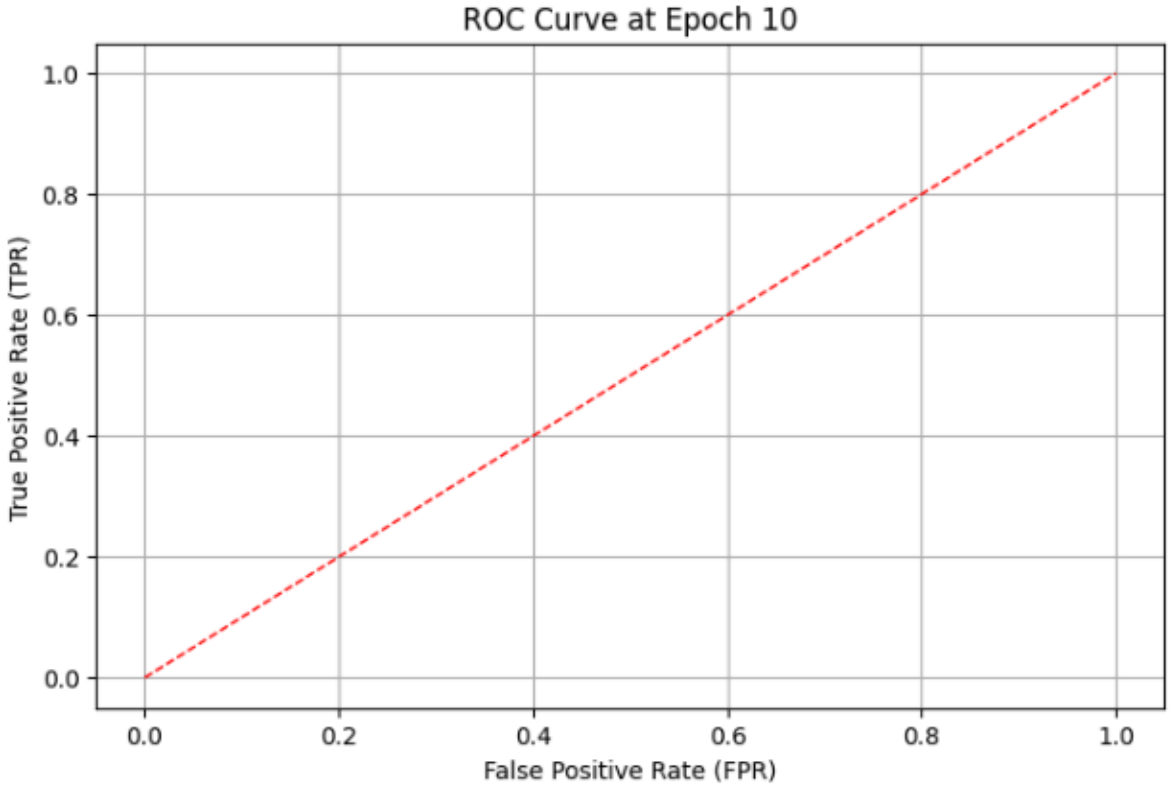
4.1 Confusion Matrix

The confusion matrix was generated for the Logistic Regression model to visualize how well the model classified each of the five classes. It helped us identify where the model struggled, especially with underrepresented classes.



4.2 ROC Curve

We generated an ROC curve for the Logistic Regression model to evaluate its performance in terms of the True Positive Rate (TPR) and False Positive Rate (FPR). The ROC curve is a valuable tool for determining the optimal threshold for classification.




```
Precision = 0.6319, Recall = 0.4969
```

---

5. Comparison of Models

We compared the performance of the three models (KNN, Logistic Regression, and Decision Tree) using key evaluation metrics such as accuracy and log loss.

Model	Accuracy	Log Loss
KNN (k=5)	0.7408	4.24

Logistic Regression	0.753 1	0.947 4
Decision Tree	0.672 6	-

5.1 Final Remarks

- **KNN** showed a greatperformance, with an optimal k=5 providing wonderful accuracy.
- **Logistic Regression** demonstrated marvelous improvement over the epochs and was the best algorithm accuracy-wise, and its final accuracy was slightly higher compared to KNN.
- **Decision Tree** performed reasonably well but did not surpass the other two models in accuracy.
- **SVM** was extremely disappointing so it was excluded from the performance competition

---

6. Conclusion

In this project, we developed an image classification model to predict the stages of diabetic retinopathy from RGB images. Multiple machine learning models, including KNN, Logistic Regression, and Decision Trees, were tested. KNN emerged as the top performer, achieving the highest accuracy. Logistic Regression, while not as accurate, showed improvement over time with iterative training. Decision Trees performed decently but did not match the performance of KNN or LR.

Moving forward, we could explore techniques to handle class imbalance more effectively, as well as refine the models through hyperparameter tuning and other advanced approaches.