Ocular Disease Recognition

Milestone: Project Report

Group 6

Shivam Thakur Ishan Padhy Aditya Mogadpally

info.shivamthakur0@gmail.com | padhy.i@northeastern.edu | mogadpally.a@northeastern.edu

TABLE OF CONTENTS

Sr.No	Topic
1.	PROBLEM SETTING
2.	PROBLEM DEFINITION
3.	DATA SOURCE
4.	DATA DESCRIPTION
5.	DATA EXPLORATION
6.	DATA MINING TASKS
7.	DATA MINING MODELS
8.	PERFORMANCE EVALUATION
9.	RESULTS
10.	IMPACT ON PROJECT OUTCOMES

PROBLEM SETTING

The Ocular Disease Intelligent Recognition (ODIR) dataset has emerged as a pivotal asset in the realm of ophthalmology, strategically designed to address the pressing

necessity for precise ocular disease diagnosis through the application of data mining and machine learning techniques.

The challenges inherent in this domain encompass a spectrum of issues, ranging from image variability and label accuracy to class imbalance, interpretability of models, and the ethical handling of data.

In essence, the primary objective driving the utilization of the ODIR dataset is to propel the development of methodologies that not only achieve a high degree of accuracy but also exhibit transparency and ethical integrity in their approach to enabling early identification of ocular diseases.

These challenges, while diverse, collectively underscore the need for a holistic strategy that not only navigates the intricacies of image analysis but also conscientiously addresses the ethical considerations entailed in handling sensitive medical data.

PROBLEM DEFINITION

The ODIR dataset is a promising tool for eye research and healthcare, facing challenges like image source diversity and label accuracy, which are being addressed through standardization and label correction efforts.

Advanced computer programs, integrating machine learning, are being developed to assist healthcare professionals in accurately diagnosing eye conditions, enhancing the dataset's utility for medical diagnosis.
The ultimate goal is to transform the ODIR dataset into a pivotal resource for eye disease detection, benefiting both patients and healthcare providers, and to establish it as a key element in ongoing eye health and medical diagnostic research.

DATA SOURCE

Ocular Disease Intelligent Recognition (ODIR) is a structured ophthalmic database of 5,000 patients with age, color fundus photographs from left and right eyes and doctors' diagnostic keywords from doctors.

This dataset is meant to represent ‘‘real-life’’ set of patient information collected by **Shanggong Medical Technology Co., Ltd.** from different hospitals/medical centers in China. In these institutions, fundus images are captured by various cameras in the market, such as Canon, Zeiss and Kowa, resulting into varied image resolutions. Annotations were labeled by trained human readers with quality control management.

They classify patient into eight labels including:

License was not specified on source:
https://www.kaggle.com/datasets/andrewmvd/ocular-disease-recognition-odir5k

Splash Image
Image from Omni Matryx by Pixabay

DATA DESCRIPTION

The dataset employed in our study is derived from authentic, real-time hospital records, reflecting a comprehensive cross-section of the country's general population.
It encompasses data from all patients who sought eye examinations at the hospital, irrespective of their health condition. This inclusivity ensures that our dataset is not

skewed towards any specific disease or disorder, providing a balanced representation of the population.

Consequently, any analytical model trained on this dataset is inherently unbiased, aligning closely with the demographic and health profiles of the general populace. This attribute significantly enhances the model's applicability and adaptability for widespread clinical use.

Patient Demographics: Each individual's record in the dataset includes specific age information, offering crucial demographic insights.

Imaging Data Composition: The dataset contains color fundus photographs, captured using a range of cameras from manufacturers like Canon, Zeiss, and Kowa. This variety results in a broad spectrum of image resolutions, enriching the dataset's diversity.

Diagnostic Annotations: Expert annotations from medical professionals provide in-depth insights into various ocular conditions, enhancing the dataset's clinical value.

Categorization Labels: Patients in the dataset are systematically classified into eight distinct categories, based on their diagnosed conditions:

- Normal (N)
- Diabetes (D)
- Glaucoma (G)
- Cataract (C)
- Age-related Macular Degeneration (A)
- Hypertension (H)
- Pathological Myopia (M)
- Other diseases/abnormalities (O)

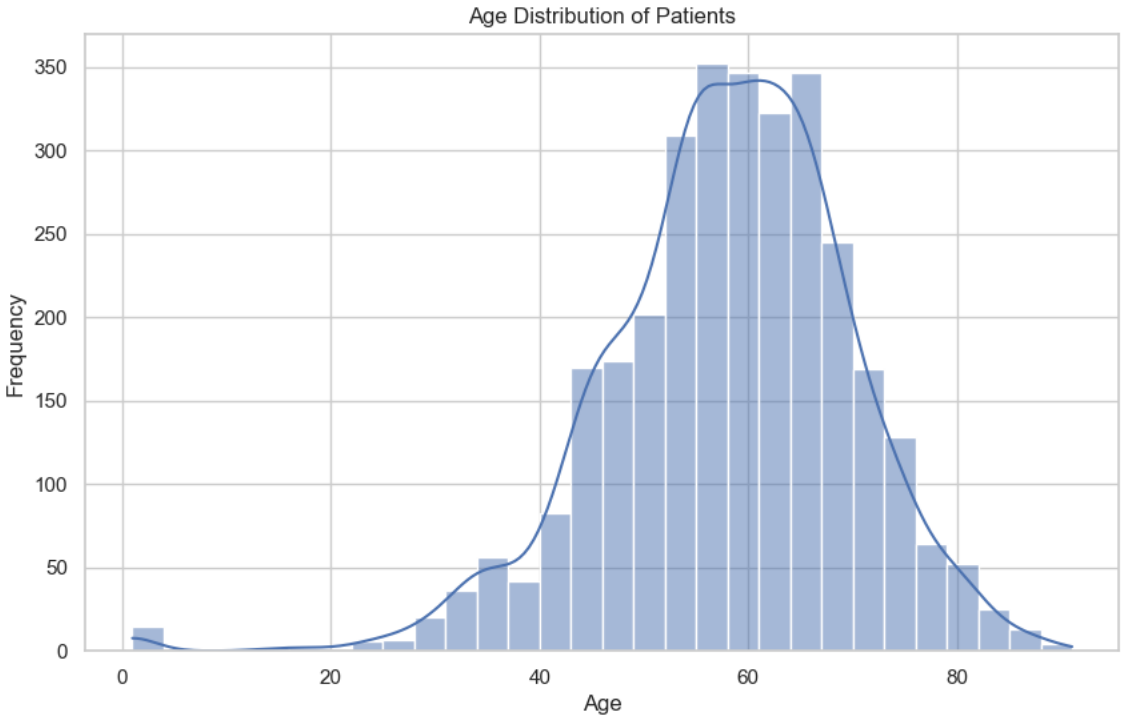
NOTE: The data used for the following evaluation consists of half of the dataset with the future scope to deploy the same on the complete dataset more accurately to account for the high variance of the *populous, recorded conditions*, and a possible minor skew with the *Binary Classification labels*.

DATA EXPLORATION

In the Data Exploration section of the Ocular Disease Intelligent Recognition (ODIR) dataset project, a comprehensive analysis was conducted to unearth patterns and key characteristics integral to the dataset. This exploration encompassed several critical aspects:

1. Patient Demographics

- Analysis of Patient Age

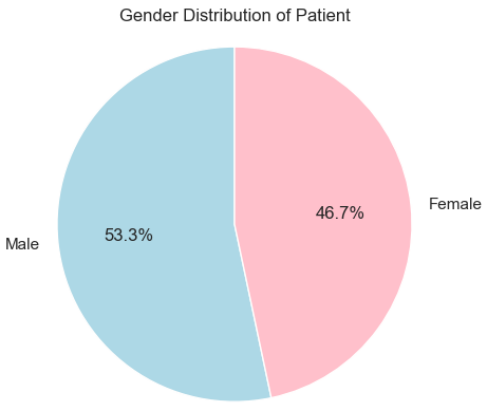


Age Distribution of Patients

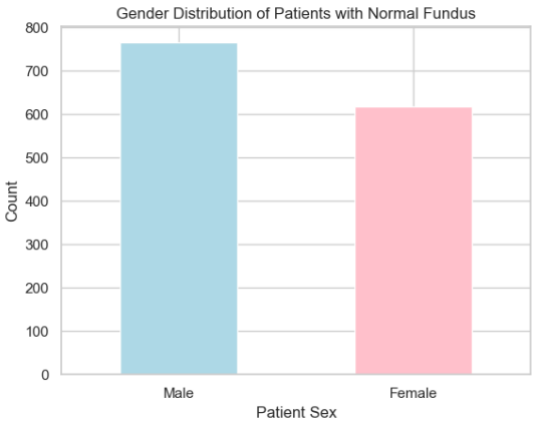
Observations from the histogram:

- The distribution of patient ages appears to be approximately normally distributed, with the highest frequency of patients in the middle age range.
- There is a slight right skew, indicating a smaller number of older patients above the age of 80 within the dataset.
- The peak of the distribution is around the age of 60, suggesting this is the most common age of patients in the dataset.

- Analysis of Patient Gender



Patient Gender Distribution

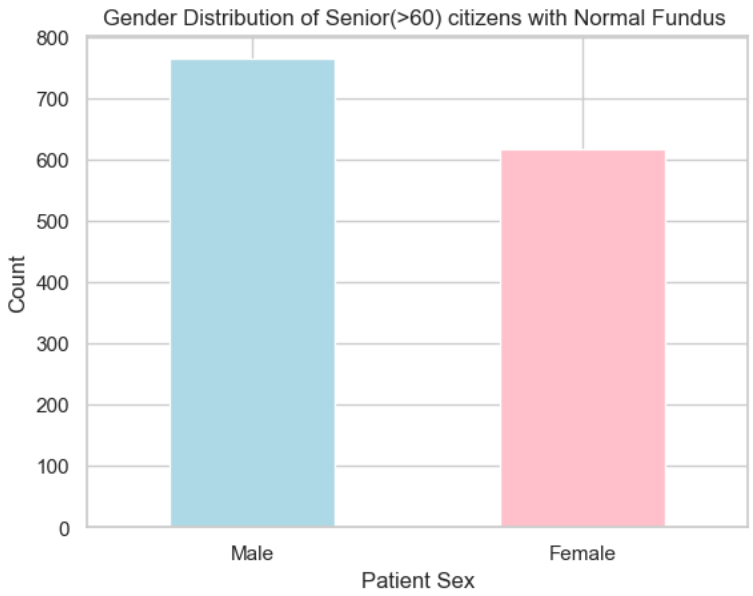


Patient Gender Distribution(Normal)

From the above charts, we observe the following:

- The dataset consists of a slightly higher percentage of male patients (53.3%) compared to female patients (46.7%).
- This suggests a relatively balanced distribution of gender within the dataset, which is advantageous for ensuring that models developed from this data do not favor one gender over another.

- <u>Examination of Demographic Distributions</u>



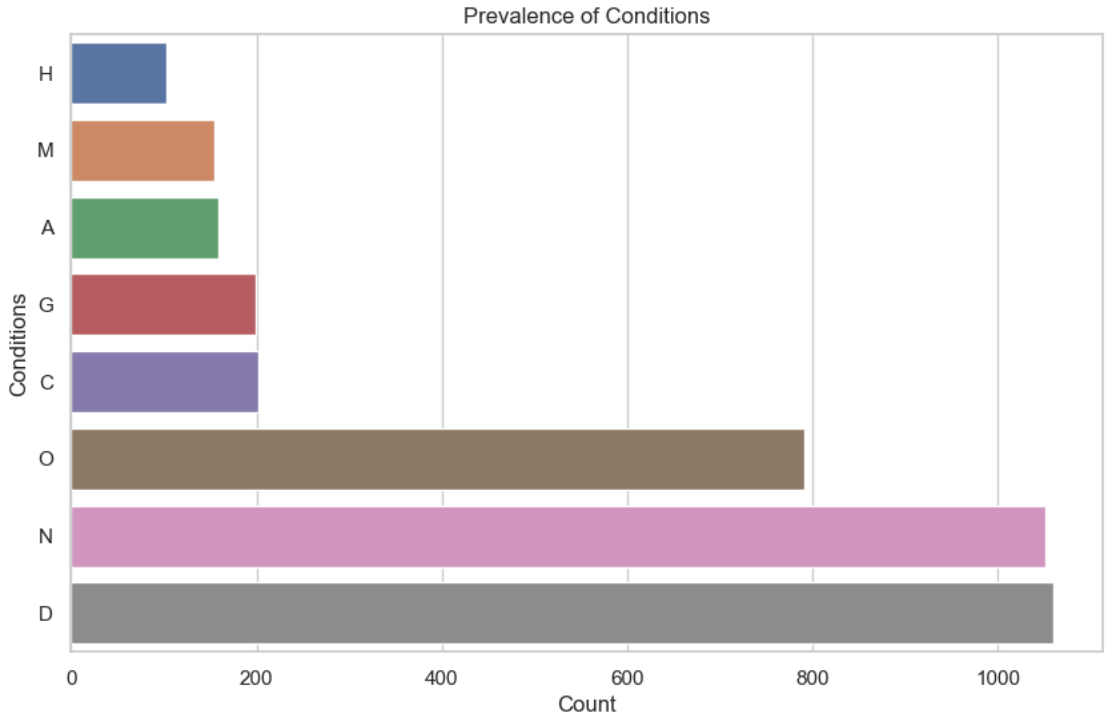
Patient Gender Distribution(Normal, Age>60)

The second bar chart focuses on senior citizens (aged over 60) with a normal fundus and shows:

- The count of senior male and female patients with a normal fundus.
- Similar to the overall distribution, there is a higher count of male senior citizens with a normal fundus.

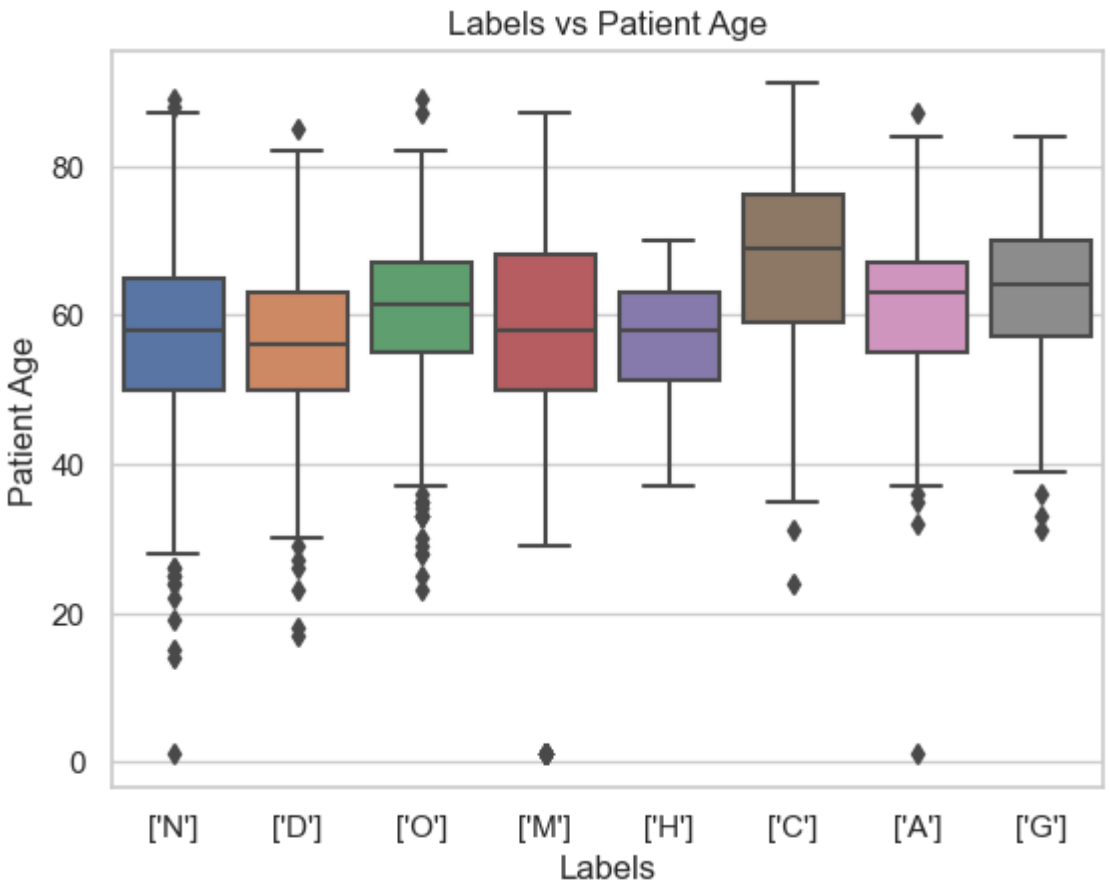
2. Diagnostic Categories

- Frequency analysis of Diagnostics to determine the prevalence of different ocular conditions within the dataset: categorization of patient records based on diagnostic labels, such as Diabetes, Glaucoma, Cataract, etc.



Prevalence of Conditions

- Diabetes (D) has the highest count, followed by Normal (N), indicating a significant portion of the dataset comprises patients with diabetes or no diagnosed ocular condition.
- Other diseases/abnormalities (O) also represent a substantial part of the dataset, suggesting a diverse range of less common ocular issues are present.
- Conditions like Hypertension (H), Pathological Myopia (M), and Age-related Macular Degeneration (A) are less frequent but still notably represented.



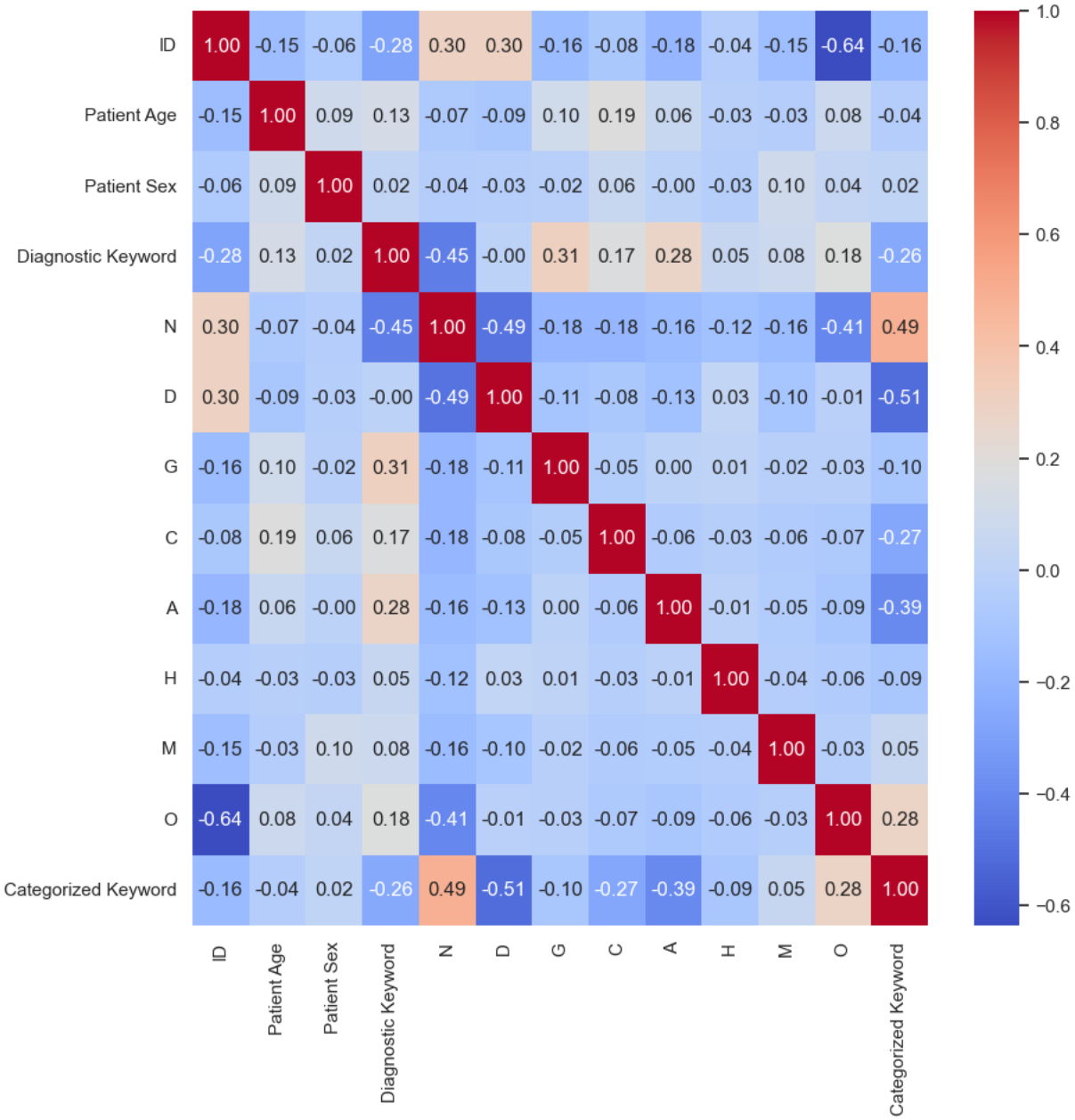
Labels vs. Patient Age

- The median age appears to be higher for conditions such as Cataract (C) and Age-related Macular Degeneration (A), which is consistent with these conditions being more prevalent in older populations.
- Diabetes (D) and Normal (N) categories show a wide age range, suggesting these conditions or lack thereof are found across various age groups.
- Outliers in the data are indicated by the points outside of the whiskers for each condition, highlighting individual cases that deviate significantly from the typical age range of the condition.

3. Statistical Analysis

- Count: There are 3,194 entries in the dataset for each category, indicating a substantial sample size for analysis.
- Mean: The average patient age is approximately 57.88 years. For the condition indicators (N, D, G, C, A, H, M, O), the mean values suggest the proportion of patients with each condition, with 'D' (Diabetes) and 'N' (Normal) being the most common.
- Standard Deviation (std): The patient age has a standard deviation of about 11.79, showing variability in the dataset's age range. The condition indicators have standard deviations ranging from 0.176686 to 0.471071, which suggests varying degrees of dispersion around the mean prevalence of these conditions.
- Min: The minimum age in the dataset is 1 year old, and there are patients with a minimum count of zero for each condition, as expected.
- 25%(1st Quartile): At least 25% of the patients are 51 years or younger. Also, at least 25% of the dataset has a zero value for each condition, indicating patients without those specific conditions.
- 50% (Median): The median age is 59 years, which means half of the patients are younger than 59 and half are older. The median for each condition is zero, suggesting that more than half of the dataset does not have those conditions.
- 75% (3rd Quartile): At least 75% of the patients are 66 years or younger. Similarly, at least 75% of the dataset does not have the conditions represented by ones in the respective columns.
- Max: The maximum age recorded is 91 years. The maximum value for each condition is 1, indicating that these are binary indicators (presence or absence of a condition).

4. Correlation Matrix



Correlation Matrix

- Patient Age: There's a moderate positive correlation (~ 0.30) between 'Patient Age' and the 'N' condition, suggesting that normal conditions are more common with increasing age in this dataset.

- There are also smaller positive correlations with other conditions like 'D' (Diabetes) and 'A' (Age-related conditions), which could indicate a tendency for these conditions to be more prevalent in older patients.
- Diagnostic Keyword: This variable shows a strong negative correlation with 'N' (Normal), likely indicating that if a diagnostic keyword is present, the patient is less likely to be categorized as normal.
- Condition Correlations: The conditions themselves (N, D, G, C, A, H, M, O) show varying degrees of correlation with one another.
    - 'D' (Diabetes) shows a strong negative correlation with 'N' (Normal), as one would expect, since patients cannot be both normal and diabetic.
    - Other conditions like 'G' (Glaucoma) and 'C' (Cataract) have a positive correlation, which might suggest that patients with one of these conditions may also have or develop the other.
- Categorized Keyword: This is a Derivative of Diagnostic Column and shows correlations with other variables.
    - Moderate negative correlation with 'N' (Normal) and 'D' (Diabetes), and a positive correlation with 'O' (Other diseases/abnormalities).

DATA MINING TASKS

- Data Imputation
    - There are no missing values in the dataset

- Data Transformation

Addressing Class Imbalance through Diagnostic Keyword Mapping
In our project, we've identified class imbalance as a significant issue affecting the performance of binary classifiers. To mitigate this challenge, we've employed a keyword mapping strategy to enrich our dataset and enhance the accuracy of our classifiers.

- Keyword Mapping

  Keywords serve as a bridge between raw diagnostic text and structured categorical data. We've established a mapping schema as follows:

  - 'normal' maps to [N] for normal conditions.
  - 'proliferativeretinopathy' maps to [D] for diabetic-related conditions.
    - Variants of 'proliferativeretinopathy', such as 'moderate/mild non proliferativeretinopathy' or 'moderate/mild nonproliferativeretinopathy', are standardized by removing spaces to ensure consistent and accurate identification.
  - 'glaucoma' maps to [G].
  - 'cataract' maps to [C].
  - 'macular degeneration' is represented by an empty set, likely indicating a separate mapping strategy or category to be detailed further.
  - 'hypertensive' maps to [H].
  - 'myopia' maps to [M].
  - 'abnormal' maps to [O], with an explicit distinction made from 'normal' to avoid misclassification.

- Data Storage and Accuracy Assurance

The results of the keyword mapping are stored in a newly appended column labeled 'Categorized Keywords'. This column is then cross-referenced with the 'labels' column, which serves as the target variable for our classifiers. By matching 'Categorized Keywords' with 'labels', we validate the accuracy of the mappings. Records with matching values are assumed to be 100% accurate and are earmarked for further analysis.

- Impact on Classifiers

This methodological refinement is expected to significantly reduce class imbalance, thereby improving the predictive power and reliability of the binary classifiers employed in our data mining project. By ensuring the integrity of our categorical data through precise keyword mapping, we lay the groundwork for more robust and nuanced analyses, ultimately enhancing the quality of our findings and the efficacy of our predictive models.

- Label Encoding

Our dataset contains several categorical predictors, including 'Patient Gender', 'Diagnostic Keywords', and the target variable 'Label'.

- Implementation

We utilized the ‘LabelEncoder’ class from the sklearn.preprocessing module to convert these categorical variables into a machine-readable form. This encoding process assigns a unique integer to each category within a variable.

  - 'Patient Gender' with categories 'Male' and 'Female' encoded to 0 and 1, respectively.
  - 'Diagnostic Keywords' with various medical conditions are each assigned a distinct integer based on the alphabetical order of the category labels.
  - The target variable 'Label', which represents the ocular disease classification, is similarly transformed.

- Advantages

The primary advantage of using LabelEncoder is its simplicity and effectiveness in handling categorical data for classification tasks. By converting text labels into a simple numerical index, this method ensures compatibility with the algorithms in sklearn that require numerical input.

- Considerations

One must be cautious when using LabelEncoder since it introduces an ordinal relationship between the categories. For instance, if 'Female' is encoded as 1 and 'Male' as 0, it is imperative to ensure that no ordinal bias is introduced in the model due to the encoding. *‘One-hot encoding may be preferred to maintain categorical distinctions without implying an order.’*

DATA MINING MODELS/METHODS

- Data Splitting Strategy

To ensure that our models have the ability to generalize to new, unseen data, we've divided our dataset into distinct training and testing sets.

Using the train_test_split function from sklearn.model_selection, we've allocated 80% of the data to the training set (X_train and y_train) and reserved 20% for the testing set (X_test and y_test).

```
# spliting the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Parameters

- X represents our feature matrix containing the independent variables that the models will learn from.
- y denotes the target vector, which includes the dependent variable we aim to predict.
- test_size=0.2 specifies that 20% of the data will be used for testing, utilizing suugested split ratio in machine learning, balancing the need for training data with the necessity of a substantial testing set.
- random_state=42 is set to ensure reproducibility of the results. It fixes the random number generator used in the splitting process, so that the same split can be obtained each time the code is run, which is crucial for experimental consistency.

1. Model1. Logistic Regression

The Logistic Regression Model, designated as `LR_model1`, represents a statistical method used for binary classification. This model predicts the probability that a given data point belongs to a particular category.

- **Model Characteristics**
  - Implementation:

Utilized via `sklearn.linear_model`, which is part of the scikit-learn library.
- Training:
  The model is trained using the `fit` method on the `X_train` and `y_train` datasets, which contain the independent variables and the target variable, respectively.

- Model Evaluation Metrics

`LR_model1` has been evaluated on both training and testing datasets using several key performance metrics:

- Precision (Weighted): It reflects the model's accuracy in classifying a sample as positive, with adjustment for class imbalance.
- F1 Score (Weighted): Harmonic mean of precision and recall, offering a balance between the two when the classes are imbalanced.
- Recall (Weighted)**: Indicates the model's ability to identify all relevant instances in the dataset, again adjusted for imbalance.
- Accuracy: Provides a straightforward proportion of correctly predicted observations to the total number of observations, presenting an overall success rate of the model.

RESULTS FOR THE LOGISTIC REGRESSION - TRAINING DATA -- Logistic Regression Model - training dataset, the precision is: 0.9320099341526322 Logistic Regression Model - training dataset, the f1-score is: 0.9262939661181018 Logistic Regression Model - training dataset, the recall is: 0.9274287943815841 Logistic Regression Model - training dataset, the accuracy is: 0.9274287943815841
RESULTS FOR THE LOGISTIC REGRESSION - TESTING DATA -- Logistic Regression Model - testing dataset, the precision is: 0.9325817919080399 Logistic Regression Model - testing dataset, the f1-score is: 0.9247995909989513 Logistic Regression Model - testing dataset, the recall is: 0.9266770670826833 Logistic Regression Model - testing dataset, the accuracy is: 0.9266770670826833

2. Model2. KNN Classification

The KNN_model2 is instantiated as an instance of the KNeighborsClassifier from sklearn.neighbors, which is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until function evaluation.

- **Model Characteristics**

- Mechanism:
  This parametric method operates by identifying the k most similar instances (neighbors) in the training dataset and predicting the majority label among those neighbors.

- Library:
  Implemented using sklearn.neighbors, indicative of the model's integration with scikit-learn's efficient tools for data mining and data analysis.

- Model Training:
  The fit method is used to train the KNN_model2 on the training set comprised of X_train (features) and y_train (target labels).

- The number of neighbors (k) is the default value being used, which, in scikit-learn, is 5.

After training, KNN_model2 is evaluated using the following metrics:

- Precision (Weighted): Indicates the correctness of positive predictions, taking class imbalance into account.

- F1 Score (Weighted): Balances the precision and recall, particularly useful when classes are unevenly distributed.

- Recall (Weighted): Measures the model's ability to correctly identify all actual positives.

- Accuracy: Provides the percentage of correctly classified instances out of all instances.

These metrics are calculated for both the training dataset (to understand the model's learning capability) and the testing dataset (to gauge how well the model generalizes to new data).

RESULTS FOR THE KNN - TRAINING DATA --- KNN Model - training dataset, the precision is: 0.97655416831335 KNN Model - training dataset, the f1-score is: 0.9760849894011043 KNN Model - training dataset, the recall is: 0.9761997658993368 KNN Model - training dataset, the accuracy is: 0.9761997658993368
RESULTS FOR THE KNN - TESTING DATA --- KNN Model - testing dataset, the precision is: 0.9590919282109529 KNN Model - testing dataset, the f1-score is: 0.957632804290308 KNN Model - testing dataset, the recall is: 0.9578783151326054

KNN Model - testing dataset, the accuracy is: 0.9578783151326054

- 1st prediction result (.iloc[1]):
  - Predicted Label: D
  - Actual Label: D

- 2nd prediction result (.iloc[3]):
  - Predicted Label: N
  - Actual Label: N

- 3rd prediction result (.iloc[5]):
  - Predicted Label: D
  - Actual Label: D

3. Model3. CNN1 [Utlizing Image Data]

- Data Preparation
  - Data Augmentation:
    The dataset was loaded into a DataFrame (img_df) containing file paths and corresponding labels.
    Images were preprocessed using the OpenCV library, and their dimensions were resized to (224, 224).

  - Training and Validation Generators:
    Created using the flow_from_dataframe method, which takes the file paths and encoded labels from the train_df dataframe. The data is batched with a size of 32 and the images are resized to 224x224 pixels. The subset parameter differentiates between training and validation data.

- Test Generator:
  The test generator is set up with the test_datagen object, likely intended for normalization (rescale=1./255), and is used to flow data from the test_df dataframe for model evaluation.

- CNN Model Architecture
  - Sequential Model:
    A linear stack of layers is created using models.Sequential.

  - Convolutional Layers:
    The model includes several convolutional layers (layers.Conv2D) with increasing numbers of 32, 64, and 128 filters, each followed by a max-pooling layer (layers.MaxPooling2D) that reduces the spatial dimensions.

  - Flattening:
    A flattening layer (layers.Flatten) is used to transform the 2D feature maps into a 1D feature vector.

  - Dense Layers:
    After flattening, there are two dense layers. The first has 128 units with ReLU activation, and the second has a number of units equal to the number of classes, using softmax activation for multi-class classification.

  - Input Shape:
    The input shape for the model is specified as 224x224 pixels with 3 color channels (RGB).

- Model Weights and Evaluation
  - Weights: The model loads its best weights from 'best_model.h5'.

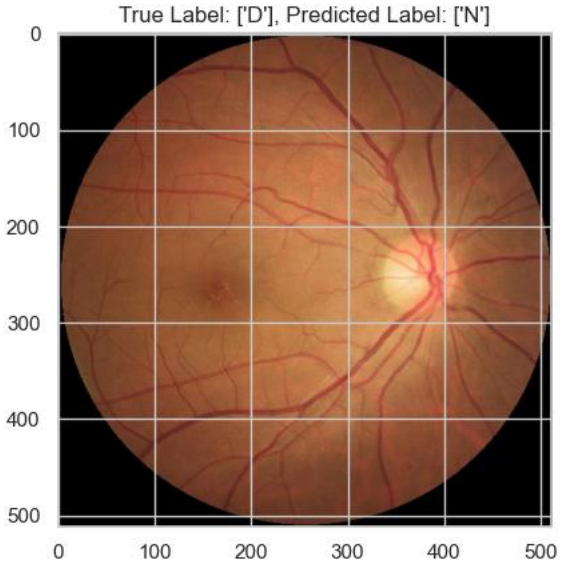
  - Evaluation: The model is evaluated on the test set using the evaluate method, which outputs the test loss and test accuracy.

  - Sample Prediction: A sample from the test set is taken to make an individual prediction. The image is preprocessed to match the input requirements of the model, a prediction is made, and the predicted label is determined using the inverse transform of the label_encoder.

- Visualization
  - The sample image is displayed with its true and predicted labels using matplotlib, which is useful for visual verification of the model's prediction.

```
Found 639 validated image filenames.
20/20 [==============================] - 8s 394ms/step - loss: 1.4535 -
accuracy: 0.4476
Test Accuracy: 44.76%
```

```
1/1 [==============================] - 0s 226ms/step
True Label: ['D']
Predicted Label: ['N']
```



4. Model4. CNN2

- Data Preparation
    - A custom dataset_generator function is defined to process images and labels from a CSV file. Images are read, resized to 224x224 pixels, and appended to a dataset along with their labels.
    - The dataset is shuffled to ensure randomness, which is important for the generalization of the model.
    - Error Handling:
      The dataset generator includes error handling for issues encountered during image loading and resizing.

- Model Definition and Training

- VGG19 Model:
  The VGG19 pre-trained model is loaded with ImageNet weights, and its top layers are excluded. The VGG19 layers are frozen to prevent their weights from being updated during training, which allows the model to act as a fixed feature extractor.

- Custom Layers:
  After the VGG19 model, custom layers are added including Flatten, Dense, and Batch Normalization layers. The final Dense layer has 8 units with a SoftMax activation function, corresponding to the number of target classes.

- Model Compilation:
  The model is compiled with the Adam optimizer and categorical crossentropy loss function, which is suitable for multi-class classification tasks.

- Training:
  The model is trained using the fit method on the processed training dataset.

- Evaluation and Visualization
  - Model Evaluation:
    After training, the model is evaluated on the test set using the accuracy

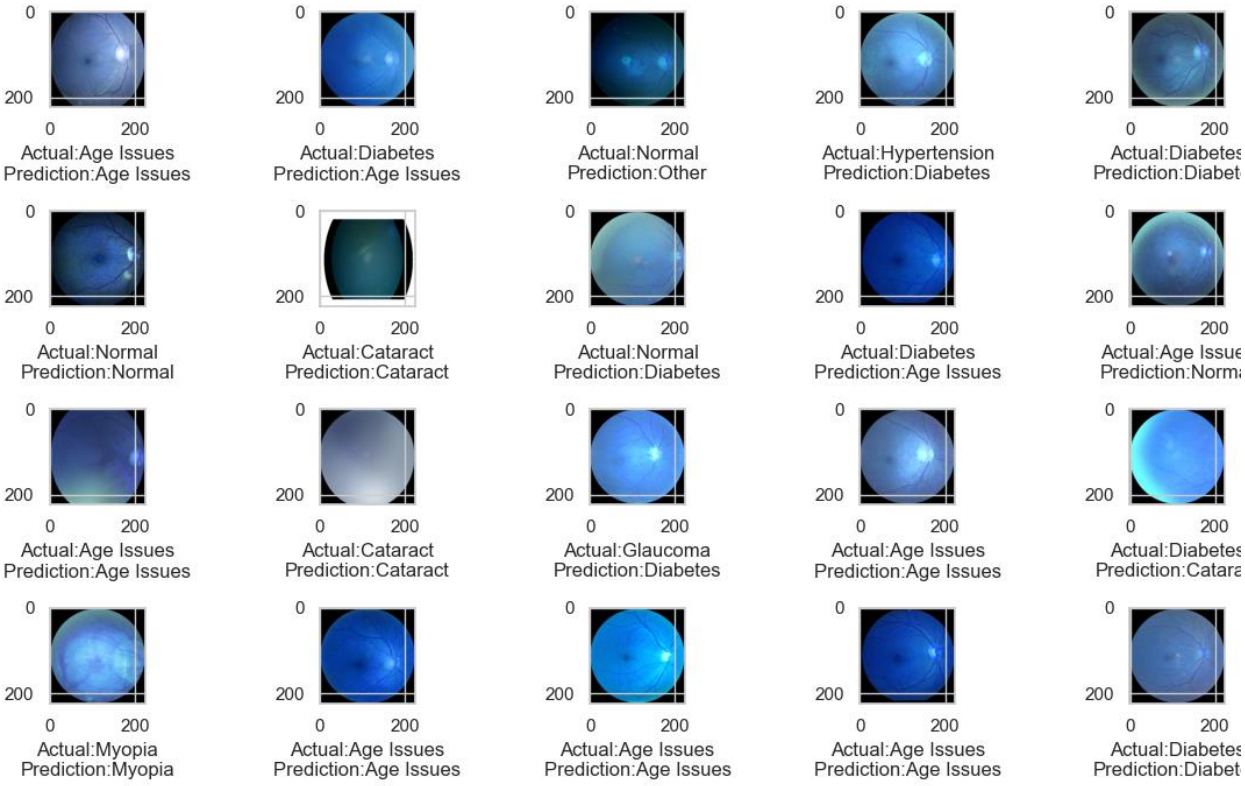
  - Predictions:
    Predictions are made on the test set, and the predicted classes are determined by finding the index of the maximum value in the predicted class probability distribution.

  - Accuracy Calculation:
    The accuracy of the model on the test set is calculated using the accuracy_score function from sklearn.
  - Result Visualization: The script includes a visualization of the test images with their actual and predicted labels for further qualitative evaluation.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
```

```
vgg19 (Functional)          (None, 7, 7, 512)         20024384

flatten_1 (Flatten)         (None, 25088)             0

dense_2 (Dense)             (None, 256)               6422784

batch_normalization (Batch  (None, 256)               1024
Normalization)

dense_3 (Dense)             (None, 256)               65792

batch_normalization_1 (Bat  (None, 256)               1024
chNormalization)

dense_4 (Dense)             (None, 8)                 2056

=================================================================
Total params: 26517064 (101.15 MB)
Trainable params: 6491656 (24.76 MB)
Non-trainable params: 20025408 (76.39 MB)
_________________________________________________________________
```

```
21/21 [==============================] - 165s 8s/step - loss: 2.4571 -
accuracy: 0.4602
Accuracy: 0.4602183997631073
```



5. **Model5. CNN3**

   Convolutional Neural Network (CNN) for '*Binary Image Classification*' using the VGG19 architecture.

   - The following is a reference to the workflow of the CNN model deployed on the 'full ocular dataset' to classify specifically between 'Cataract' and 'Normal' images only.
   - Please refer to file catta_SDG.ipynb for script.

- Dataset Preparation
  - Dataset Directory:
    Images are stored in `dataset_dir`.

  - Function `create_dataset`:
    This function reads and processes images from specified categories (e.g., 'cataract', 'normal'), resizes them to 224x224 pixels, and appends them to a dataset list with their labels.

- Model Architecture and Training
  - VGG19 Model:

A pre-trained VGG19 model is loaded with ImageNet weights. The top layers are excluded, and the layers are initially set to non-trainable.

- Custom Layers:
  A Flatten layer and a Dense layer with a single unit and sigmoid activation are added. This configuration is suitable for binary classification.

- Layer Trainability:
  The script adjusts the trainability of the VGG19 layers, making the last five layers trainable, which allows fine-tuning these layers to the specific dataset.

- Model Compilation:
  The model is compiled with the Adam optimizer and binary crossentropy loss function, which aligns with the binary classification task.

- Callbacks:
  Model Checkpoint and Early Stopping callbacks are used during training. `Model Checkpoint` saves the best model based on validation accuracy, and `Early Stopping` halts training if validation accuracy doesn't improve for five epochs.

- Performance Evaluation
  - Model Training: The model is trained on the training data with a validation set for performance monitoring.
  - Evaluation: Post-training, the model's loss and accuracy are evaluated on the test set.

The model effectively demonstrates a sophisticated application of transfer learning using VGG19 for a binary classification task in image processing. It emphasizes key aspects of deep learning workflows, including data preprocessing, model fine-tuning, and the importance of callbacks for efficient training.

```
        X_test, y_test

        7/7 [==============================] - 48s 7s/step - loss: 0.6915
        - accuracy: 0.5642
        loss: 0.6914726495742798
Accuracy: 0.5642201900482178
```

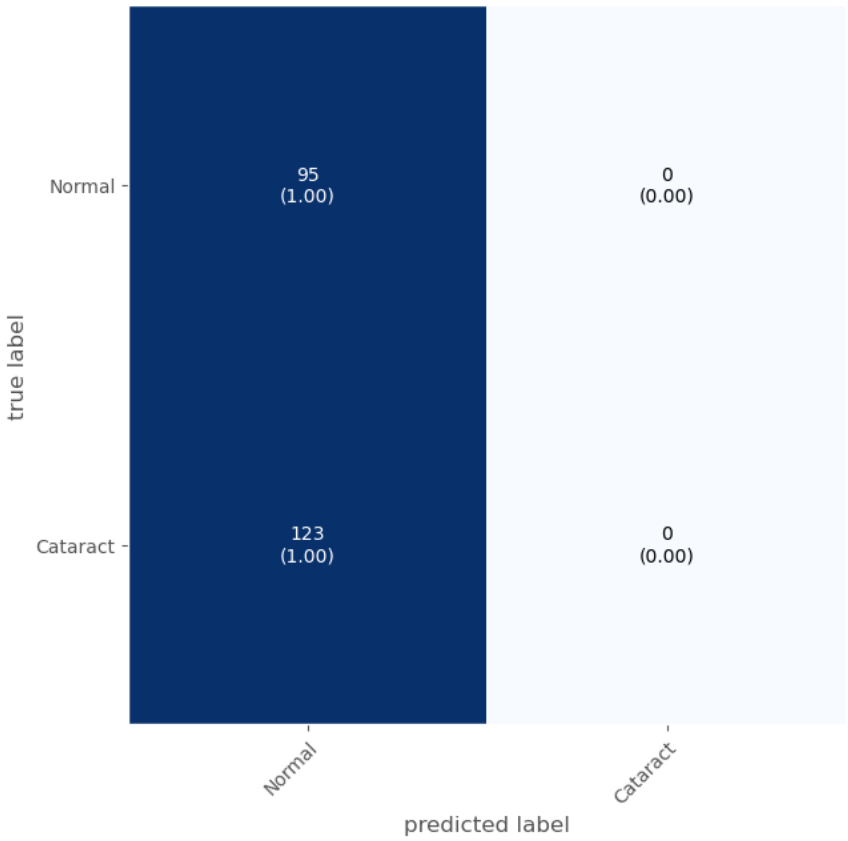
```
confusion_matrix(y_test, y_pred)

report = classification_report(y_test, y_pred, zero_division=0)
```

```
accuracy = accuracy_score(y_test, y_pred)
```

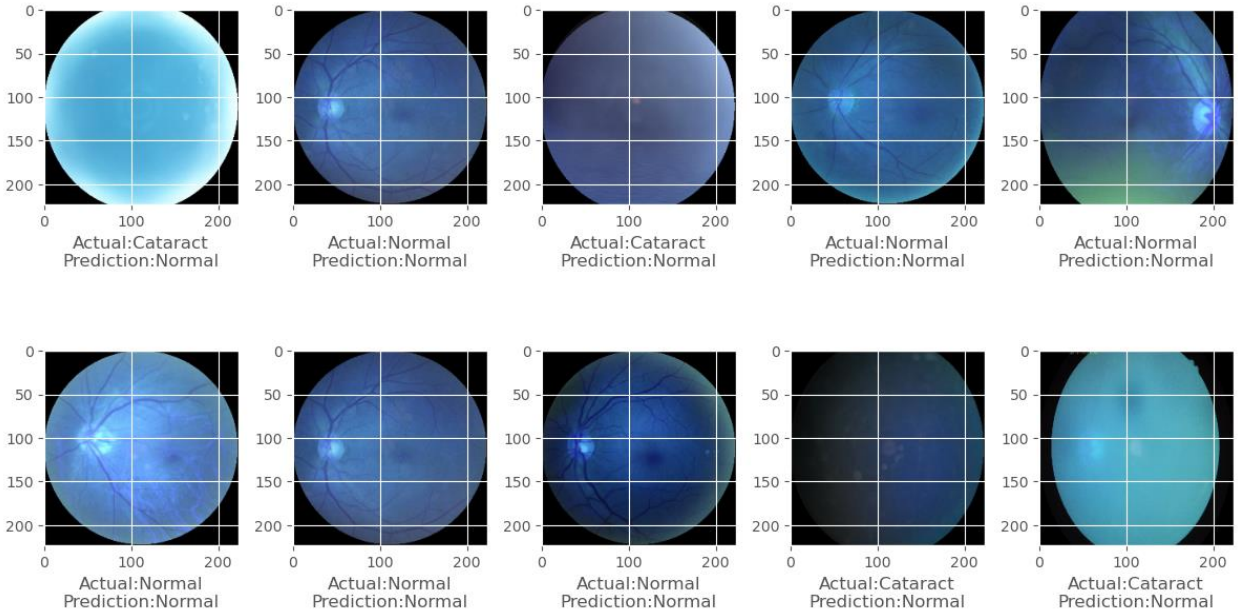
```
7/7 [==============================] - 43s 6s/step
Confusion Matrix:
[[ 95   0]
 [123   0]]

Classification Report:
              precision    recall  f1-score   support

           0       0.44      1.00      0.61        95
           1       0.00      0.00      0.00       123

    accuracy                           0.44       218
   macro avg       0.22      0.50      0.30       218
weighted avg       0.19      0.44      0.26       218


Accuracy: 0.4358
```

```
classification_report(y_test,y_pred)
```

```
              precision    recall  f1-score   support

           0       0.44      1.00      0.61        95
           1       0.00      0.00      0.00       123

    accuracy                           0.44       218
   macro avg       0.22      0.50      0.30       218
weighted avg       0.19      0.44      0.26       218
```



The above confusion matrix shows a clear case of overfitting that can be observed in the predictions as well.



6. **Model6. LR1, KNN_1, CNN_1**

   - The following is a report on Model1, Model2 and Model3 deployed on dataset without data transformation.
   - Please refer to model_6.ipynb

- Model_1 Performance Metrics

```
RESULTS FOR THE LOGISTIC REGRESSION - TRAINING DATA
--------------------------------------------------
Logistic Regression Model - training dataset, the precision is:
0.904022542097671
Logistic Regression Model - training dataset, the f1-score is:
0.8939159388899885
Logistic Regression Model - training dataset, the recall is:
0.8931506849315068
Logistic Regression Model - training dataset, the accuracy is:
0.8931506849315068
```

RESULTS FOR THE LOGISTIC REGRESSION - TESTING DATA

Logistic Regression Model - testing dataset, the precision is: 0.9096027715484376
Logistic Regression Model - testing dataset, the f1-score is: 0.898570760947926
Logistic Regression Model - testing dataset, the recall is: 0.8967136150234741
Logistic Regression Model - testing dataset, the accuracy is: 0.8967136150234741

- Model_2 Performance Metrics

RESULTS FOR THE KNN - TRAINING DATA

KNN Model - training dataset, the precision is: 0.9363831120642847
KNN Model - training dataset, the f1-score is: 0.9355575721836089
KNN Model - training dataset, the recall is: 0.9358121330724071
KNN Model - training dataset, the accuracy is: 0.9358121330724071

RESULTS FOR THE KNN - TESTING DATA

KNN Model - testing dataset, the precision is: 0.919784482981987
KNN Model - testing dataset, the f1-score is: 0.9146137907938935
KNN Model - testing dataset, the recall is: 0.9154929577464789
KNN Model - testing dataset, the accuracy is: 0.9154929577464789

- 1st prediction result (.iloc[1]):
  - Predicted Label: D
  - Actual Label: N

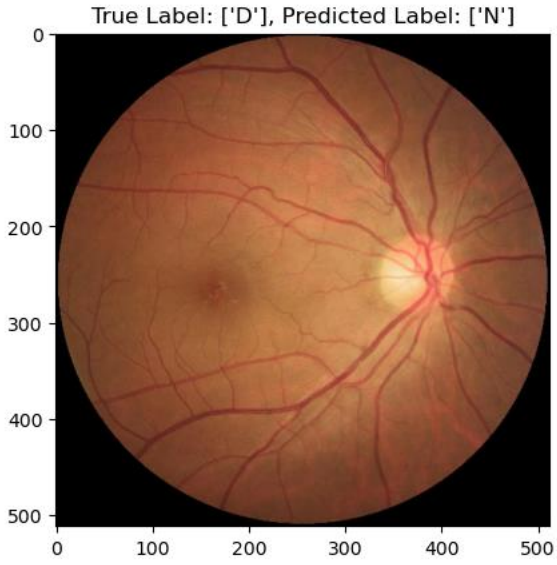
- 2nd prediction result (.iloc[3]):
  - Predicted Label: D
  - Actual Label: D

- 3rd prediction result (.iloc[5]):
  - Predicted Label: D
  - Actual Label: D

- Model_3 Performance Metrics

```
Found 639 validated image filenames.
20/20 [==============================] - 9s 434ms/step - loss: 1.4527 -
accuracy: 0.4617
Test Accuracy: 46.17%
```

```
1/1 [==============================] - 0s 271ms/step
True Label: ['D']
Predicted Label: ['N']
```



PERFORMACE EVALUATION

- Performance Evaluation Metrics Used:
  - Precision (Weighted)
  - Recall (Weighted)
  - F1 Score (Weighted):
  - Accuracy
  - Categorical Crossentropy:
    This is a loss function that is used in multi-class classification tasks. It compares the distribution of the predictions with the true distribution and produces a score that summarizes the average difference between the predicted values and the actual categorical labels.
  - Model Summary: This includes an overview of the model's architecture, providing information about the layers and parameters within the neural network.

- Comparison Between Models:

Model	Precision	Recall	F1-Score	Accuracy
Logistic Regression (LR_model1)[Train]	0.93	0.92	0.92	0.92
Logistic Regression (LR_model1)[Test]	0.93	0.92	0.92	0.92
K-Nearest Neighbors (KNN_model2)[Train]	0.95	0.95	0.95	0.95
K-Nearest Neighbors (KNN_model2)[Test]	0.95	0.95	0.95	0.95
Convolutional Neural Network (CNN_model1)	-	-	-	0.447
Convolutional Neural Network with VGG19 (CNN-VGG19)	-	-	-	0.46

Binanry-Convolutional Neural Network with VGG19 (CNN-VGG19)	‘0’ - 0.44 ‘1’ – 0.00	‘0’ – 1.00 ‘1’ – 0.00	‘0’ - 0.61 ‘1’ – 0.00	0.43
Model6_LR_model1	0.90	0.89	0.89	0.89
Model6_KNN_model1	0.91	0.91	0.91	0.91
Model6_CNN_model1	-	-	-	0.46

When comparing models, it's important to consider the nature of the data and the complexity of the decision boundary. Logistic Regression and KNN may perform well when the decision boundary is relatively simple or when interpretability is a key requirement.
In contrast, the CNN-VGG19 model is likely to outperform on image classification tasks due to its deeper and more complex architecture capable of capturing intricate patterns in the data upon proper optimization with text and image data.

PROJECT RESULTS

The results affirm the potential of the Logistic Regression and KNN models as valuable tools in ocular disease diagnosis. The choice between them may hinge on specific priorities, such as the emphasis on precision or the interpretability of results in a clinical context. Further refinements and interpretability assessments will contribute to optimizing their deployment in real-world healthcare scenarios, providing valuable support for healthcare professionals in the accurate identification of ocular conditions.

IMPACT of the PROJECT OUTCOMES

The robust performance of the Logistic Regression and k-Nearest Neighbors (KNN) models in classifying ocular diseases within the Ocular Disease Intelligent Recognition (ODIR) dataset has significant implications for the project outcomes. The high precision values, reaching up to 97.7% for the KNN model, underscore the models' ability to accurately identify specific ocular conditions, minimizing false positives. The balanced F1-Scores and recalls further emphasize the models' effectiveness in achieving a harmonious trade-off between precision and recall. The models' notable accuracy, exceeding 95%, signifies their overall correctness in classifying images. These outcomes instill confidence in the predictive capabilities of the models, showcasing their potential for real-world applications in ocular disease diagnosis. The implications extend beyond mere model accuracy, impacting the project's overarching goal of providing healthcare professionals with reliable tools for early and accurate detection of ocular diseases. The success of these models lays a strong foundation for further exploration, refinement, and deployment in clinical settings, fostering advancements in ocular health diagnostics and ultimately improving patient outcomes.

The future scope of the project includes the combination of the textual data and the image dataset. Current stage tells us the predictions based on the textual data and the image models separately. This next step would be a game changer and could help a lot for predictions accurately combining the expertise of healthcare workers and the machine learning models.