INSTITUTE OF BUSINESS ADMINISTRATION KARACHI

PROJECT REPORT

(APPLICATION DEVELOPMENT)

[FINAL PROJECT]

GROUP MEMBERS

S.NO	Name	ERP
1	JAWWAD AHMED	29423
2	MUHAMMAD GHULAM ABBAS	29417
3	MUHAMMAD UZAIR	29414

ABSTARCT

The objective of this project is the data handling, data cleaning, and data preprocessing using the Glaucoma Dataset obtained from Kaggle. The dataset reflects information related to individuals diagnosed with glaucoma, following the guidance of our instructor, we began by selecting the Glaucoma Dataset from Kaggle. Subsequently, we engaged in tasks such as data cleaning, preprocessing, and visualization to gain insights into factors influencing glaucoma diagnoses.

Contents

SYSTEM STUDY / DOMAIN ANALYSIS

DOMAIN ANALYSIS

Business Process related to Dataset.

In our exploration of glaucoma data, we aim to comprehensively analyze key features such as age, gender, eye pressure, family history, medical history, medication usage, visual test results, and visual symptoms to enhance our understanding of glaucoma diagnoses and its various types. Glaucoma, a group of eye diseases, poses the risk of vision loss and blindness by damaging the optic nerve situated at the back of the eye. Through a range of, we intend to summarize and describe the dataset's characteristics, identify patterns and relationships, pinpoint factors associated with different glaucoma types, propose interventions based on findings, offer personalized recommendations for patients, identify potential risk factors and preventive measures, detect outliers and anomalies, and comprehend the distributions of key features. This analytical approach holds immense utility for glaucoma detection and diagnosis, providing invaluable insights for medical practitioners, researchers, and healthcare professionals. The outcomes promise to inform decision-making processes, contribute to an enhanced understanding of patient patterns, and ultimately lead to improved patient outcomes, thereby advancing medical knowledge and healthcare practices related to glaucoma.

The integration of cutting-edge technologies like machine learning and artificial intelligence is increasingly prevalent in healthcare. These innovations present transformative opportunities for medical practitioners and healthcare organizations by providing accurate, data-driven insights and enabling informed decision- making. Patient-related data, including medical records, diagnoses, and treatment history, is systematically collected and stored in centralized databases after rigorous cleaning and preprocessing procedures.

Within glaucoma research and diagnosis, analyzing correlations between various patient attributes, including age, family history, medical history, and visual test results, reveals crucial patterns and indicators. This analysis, encompassing both individual patients and specific types of glaucoma, allows for a nuanced understanding of the disease. This approach facilitates the identification of areas for improvement, optimization of interventions, and fostering patient adherence.

The stored medical data serves as a valuable tool for comprehending patient behavior and patterns over time. Features like the frequency of eye pressure measurements, medication usage, and responses to visual tests provide insights into patient tendencies. By correlating this information with glaucoma diagnoses and treatment outcomes, healthcare professionals can engage in target market analysis and patient segmentation based on patient characteristics and other relevant factors. This enables them to tailor healthcare interventions and resources more effectively.

The analytical methods employed in glaucoma research encompass both descriptive and predictive analyses. Descriptive data analysis, similar to its usage in other domains, involves summarizing and interpreting the characteristics of patient data. This practice, enriched by historical patient records, enables precise insights that are pivotal for reaching optimal medical.

decisions. Key performance indicators in this context include patient age, medical history, and the effectiveness of past interventions. Correlating these factors with patient feedback allows for the derivation of preferential treatment patterns and patient satisfaction, shedding light on both tangible aspects like treatment efficacy and intangible aspects such as the quality of medical care provided.

DESCRIBING THE DATASET

Classification

The Glaucoma dataset, with columns encompassing patient demographics, medical history, and diagnostic measurements, is well-suited for classification tasks, particularly in the context of glaucoma detection. The dataset includes essential features such as age, gender, visual acuity measurements, intraocular pressure (IOP), cup-to-disc ratio (CDR), family history, medication usage, and various other diagnostic indicators. The primary objective in this context is to predict the presence or absence of glaucoma based on these diverse attributes. The classification task involves training machine learning models to learn patterns and associations within the labeled data, allowing for the accurate identification of patients with glaucoma. Common classification algorithms such as decision trees, support vector machines, or artificial neural networks could be employed to leverage the dataset's information for effective glaucoma detection. The outcome of this classification effort holds significant clinical implications, assisting healthcare professionals in early diagnosis and intervention for individuals at risk of glaucoma.

Balanced or Imbalanced Dataset?

Our dataset, with a breakdown of approximately 50.12% for Glaucoma and 49.88% for No Glaucoma, is quite balanced. This balance is crucial because if the numbers were skewed, it could impact the accuracy of our machine learning model. To better grasp the distribution of diagnoses, we utilize the expression df['Diagnosis'].value_counts(normalize=True) * 100, which provides a clear breakdown.
If the dataset were imbalanced, meaning one eye condition significantly outnumbered the other, our model might develop a preference for predicting the more common condition. This could result in a misleadingly high accuracy score, as it might excel at predicting the majority but struggle with the minority.

In such cases, accuracy alone becomes an unreliable measure. To address this, we might need to use techniques specifically designed for imbalanced datasets, ensuring that our model is trained to make accurate predictions for both Glaucoma and No Glaucoma. This way, we avoid biases and ensure a more robust and fair assessment of our model's performance.

Data Composition

The glaucoma dataset has 10000 rows and 17 columns. This dataset specifically focuses on identifying glaucoma in patients based on various factors.

The attribute names and data types are as below:

Attribute Names	Data Types	Missing Values	Feature Importance
Patient ID	int64	0	drop
Age	int64	0	High
Gender	object	0	Low
Visual Acuity Measurements	object	0	High
Intraocular Pressure (IOP)	float64	0	High
Cup-to-Disc Ratio	float64	0	High
Family History	object	0	High
Medical History	object	0	Medium
Medication Usage	object	1231	Medium
Visual Field Test Results	object	0	High
Optical Coherence Tomography (OCT) Results	object	0	High
Pachymetry	float64	0	High
Cataract Status	object	0	High
Angle Closure Status	object	0	High
Visual Symptoms	object	0	High
Diagnosis	object	0	class
Glaucoma Type	object	0	drop

We will begin by removing the Patient ID column, as it does not provide valuable insights or crucial information for analysis. Additionally, considering potential sensitivity, dropping this column safeguards patient privacy. The ' Glaucoma Type ' column will also be removed, as it does not contribute specific information.

For missing values in the 'Medication Usage' column, we will create two new features. The first feature will flag rows with no reported medication usage (null values) with a value of 1 and 0 otherwise. This allows us to readily identify patients who aren't taking any medication. The second feature will be created using Pandas' `str.get_dummies` function, generating separate columns for each distinct medication mentioned in the 'Medication Usage' column. This enables a more effective analysis of the presence or absence of each medication.

Finally, we will integrate the original Data Frame with the newly created one-hot encoded medication features using Pandas' `concat` function. This merges the information into a single, comprehensive data structure. To avoid redundancy and streamline our data, we'll remove the original 'Medication Usage' column, as its information is now captured in the one-hot encoded features."

DATA CLEANING

The data set may contain incorrect, duplicate, or erroneous values. It can also be defined as data sets containing mis ordered values, mislabeling, redundancy and other issues. In order to properly interpret and analyze data, data scientists ensure that the data is cleaned and processed properly so that it is ready for the next phase of processing and can train a model.

When you use the data.dtypes() command, all data types in the data set are checked. Given that they are correct, we proceed to the next phase of data cleaning, i.e.. we deal with lost goals.

A statistical description of the entire data set is displayed using the data.describe() command. Scanning statistical values can provide an estimate of the maximum and minimum values, thus detecting any abnormal or abnormal values.

When you use the data.isnull().sum() command, the data set is checked for null or no values. The results presented showed that there were no missing items in this data set. data[‘column name’].value_counts() can be used to check the specific classes of all columns and their value counts.

Various methods can be used to facilitate logical analysis to ensure data accuracy in order to deal with missing values and to improve data structure One method is to convert categorical columns to Boolean values or numeric representations. For example, the ‘Family History’ column can be converted to a Boolean value using np.where, where ‘Yes’ maps to True and the other values to False.

Numeric variables can be accessed using the lambda function. For example, a column labeled 'column_name' can be processed to extract statistical information. Split the result into two separate columns ('vftr_sensitivity' and 'vftr_specificity') using str.split, t

In addition to handling missing values, the 'Medication Usage' column can be leveraged to create informative features. A new column, 'No Medication,' is generated by identifying null values. If 'Medication Usage' is null, the 'No Medication' column is assigned a value of 1; otherwise, it receives a value of 0.

To capture distinct medicines, a one-hot encoding technique is applied using the str.get_dummies method. Assuming medicines are separated by commas and spaces, this process results in a new dataframe, 'medication_encoded.' The one-hot encoded medicines are then concatenated with the original dataset, enhancing the representation of medication information. Subsequently, the original 'Medication Usage' column is dropped, streamlining the dataset for further analysis.

EXPLORATORY DATA ANALYSIS

Exploratory Data Analysis (EDA) is a data analysis approach aimed at summarizing the essential characteristics of a dataset through visual methods. It involves extracting key insights from previously collected data using visualization techniques to enhance analysis. EDA is crucial for hotels to tailor their facilities and services to customer preferences, adapting existing systems for optimal user engagement and ratings. Accurate decision-making in this regard relies on the effective execution of EDA.

Exploratory Data Analysis typically employs the following methods:

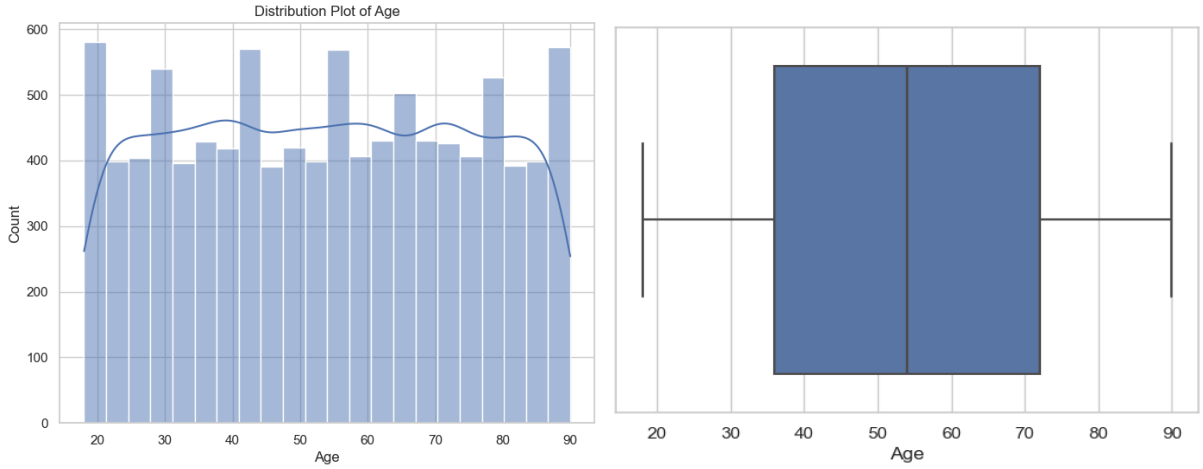
- Uni-variate analysis: Summarizes statistics for each field in the raw dataset or focuses on a single variable. Examples include histograms, box plots, pie charts, and horizontal and vertical bars.

- Bivariate analysis: Examines the relationship between each variable in the dataset and the target variable of interest, involving the analysis of two variables and their correlation. Examples include box plots, line plots, and horizontal and vertical bars.

UNIVARIATE ANALYSIS OF PROJECT:

Plots for Numerical Variables

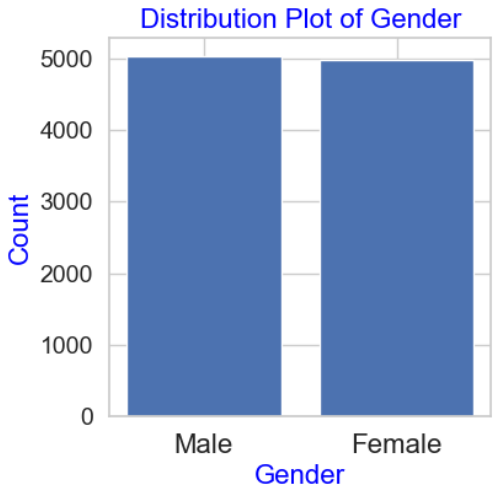
Age



Analyzing the distribution of age in the glaucoma dataset reveals an interesting phenomenon. While the histogram depicts a **uniform distribution**, indicating all ages are equally likely, the boxplot paints a different picture. It showcases a **right-skewed distribution**, meaning there's a higher concentration of individuals in the younger age groups. This seemingly contradictory information can be reconciled by considering the specific characteristics of each plot.

The flat histogram in the distribution plot suggests that within the collected data range, all age values have an equal chance of appearing. However, the boxplot delves deeper, revealing a clustering of individuals towards younger ages within this seemingly uniform distribution. These clustering manifests as the box being shifted to the left and the whisker extending further to the right, highlighting the presence of younger individuals.

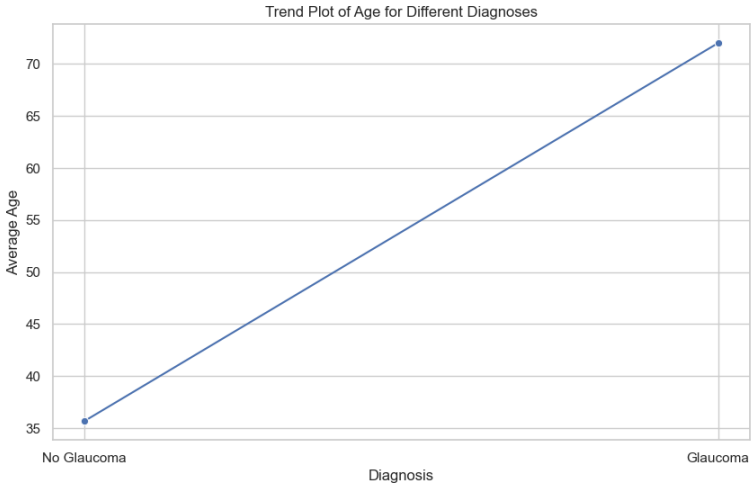
Gender



The bar plot shows that the distribution of gender is uneven, with more females (1000) than males (500) in the dataset. This is evident from the taller bar for females.

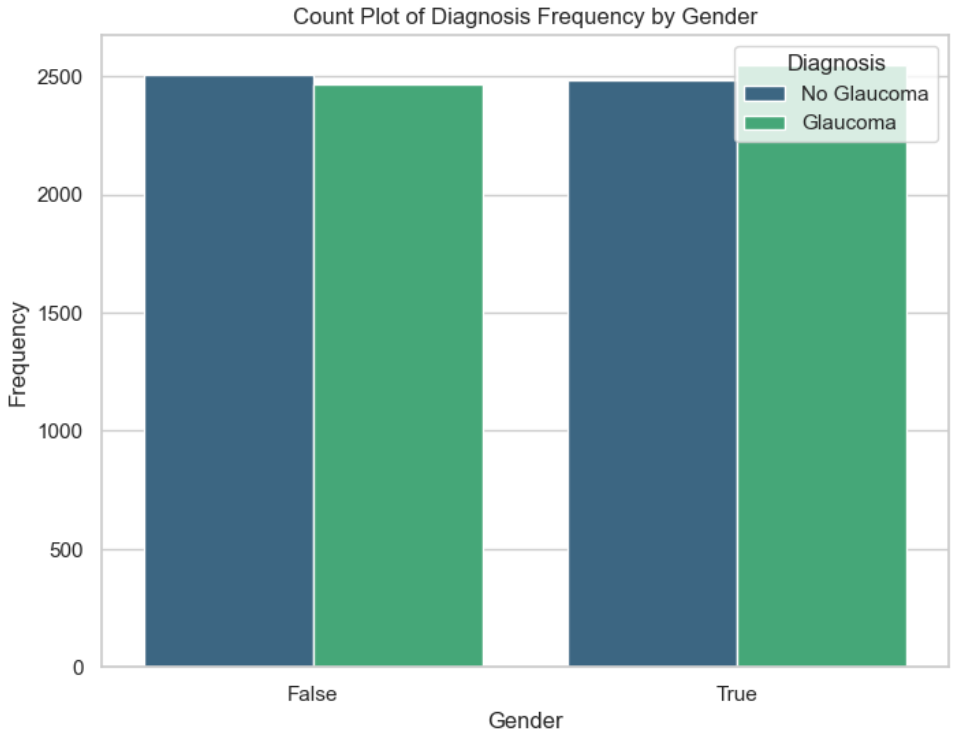
BIVARIATE ANALYSIS

Average Age VS Glaucoma



The plot shows that there is a general trend of increasing age with Glaucoma. This is evident from the upward slope of the line.

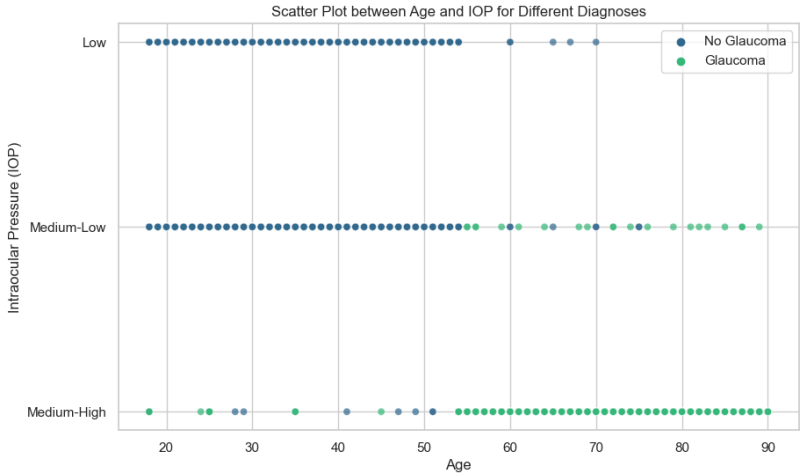
Diagnosis Frequency by Gender



The count plot shows the frequency of diagnosis by gender. The plot shows that females are more likely to be diagnosed with Glaucoma than males.

TRIVARIATE ANALYSIS

Scatter Plot between Age and IOP for Different Diagnoses



The plot shows a general trend of increasing IOP with age. This is evident from the upward slope of the point cloud. However, there is also some variation in IOP for different diagnoses. For example, the point cloud for Glaucoma is shifted to the right, indicating that people with Glaucoma tend to have higher IOP than people without Glaucoma.

DATA PREPROCESSING

Preprocessing is one of the most important steps in any machine learning task. Preprocessing basically means cleaning the text and preparing it for input into the model. There are many useful techniques when it comes to data preprocessing.

Since the data consists of both numerical and categorical values, pre-processing should be done accordingly to make the data ready and suitable for analytical operations, forecasting and modeling.

DISCRETIZATION

Discretization of functions is an important part of data preprocessing. Through discretization, a continuous data variable can be transformed into a discrete form. This is accomplished by creating a set of contiguous intervals that apply to the entire range of a continuous variable/column of data, making it discrete.

Simply put, discretization is the process of separating and storing variables into meaningful categories to better understand continuous data and infer the relationship and influence of those categories on a class variable. Interpreting features in correlation with continuous data is now easier as it has become a discrete set of information.

Two characteristics of numerical data are discretized.

Discretization based on Binning

'Low', 'Medium-Low', 'Medium-High', 'High'

10, 13.76

13.76, 17.485

17.485, 21.3

21.3, 25

In [53]:
```
df['Intraocular Pressure (IOP)'].describe()
```

Out[53]:
```
count    10000.000000
mean        17.507527
std          4.356101
min         10.000000
25%         13.760000
50%         17.485000
75%         21.300000
max         25.000000
Name: Intraocular Pressure (IOP), dtype: float64
```

In [54]:
```
df['Intraocular Pressure (IOP)'] = np.where((df['Intraocular Pressure (IOP)'] <13.67) ,'Low',
                                np.where((df['Intraocular Pressure (IOP)'] >=13.67) & (df['Intraocular Pressure (IOP)'] <=17
                               np.where((df['Intraocular Pressure (IOP)']>17.485) & (df['Intraocular Pressure (IOP)'] <=50),
                                        'High')))
```

In [55]:
```
df[['Intraocular Pressure (IOP)']]
```

Out[55]:

	Intraocular Pressure (IOP)
0	Low
1	Low
2	Medium-Low
3	Medium-Low
4	Low
...	...
9995	Medium-High
9996	Medium-High
9997	Medium-High
9998	Medium-High
9999	Medium-High

10000 rows × 1 columns

NORMALIZATION

The next stage of data preprocessing is normalization. This can be described as a training method that brings all numerical values into one range. Also known as feature scaling, this is a commonly used process in machine learning. If the ranges of the features in the dataset used are different, it is necessary to convert these columns of the dataset to the same scale, which is called data normalization.

There are several types of normalization techniques used in machine learning. However, when working with a dataset, it is required to follow the same normalization technique for all numerical variables/attributes.

- Z-score Normalization (Standardization Scaling)
  This technique gives the data values (scores) a common standard. It refers to the process of centering a variable at zero and standardizing the variance at one. The procedure of finding the z-score is subtracting the mean of each observation column from a chosen value and then dividing it by the standard deviation.
  The strategy of normalizing data helps avoid the issues of outliers. If a value is exactly equal to the mean of all the values of the feature, it will be normalized to 0. If it is below the mean, it will be a negative number, and if it is above the mean it will be a positive number. The size of those negative and positive numbers is determined by the standard deviation of the original feature. If the un-normalized data had a large standard deviation, the normalized values will be closer to 0.

E.2 Normalization

ZScore Normalization

In [71]:
```
df['Age'].value_counts()
```

Out[71]:
```
18    175
70    164
59    163
42    161
55    161
     ...
77    118
58    116
76    114
45    110
66    108
Name: Age, Length: 73, dtype: int64
```

In [72]:
```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
num_norm=scaler.fit_transform(df[num_col]).round(2)
```

In [73]:
```
num_norm
```

Out[73]:
```
array([[-1.7 , -1.31,  1.1 ,  1.16, -0.57,  0.07],
       [-1.7 , -1.31,  0.03, -0.35, -1.12,  0.26],
       [-1.7 , -0.06, -1.62,  1.17, -0.53, -0.72],
       ...,
       [ 1.71,  0.77, -0.  , -0.68, -0.48, -0.08],
       [ 1.71,  1.6 , -1.62,  1.26,  0.01,  1.42],
       [ 1.71,  0.63,  0.14, -1.31,  0.32, -1.6 ]])
```

In [74]:
```
df_num_norm=pd.DataFrame(num_norm, columns=df[num_col].columns)
```

In [75]:
```
df_num_norm
```

Out[75]:

Out[75]:

	Age	Cup-to-Disc Ratio (CDR)	oct_retinal	oct_macular	oct_rnfl	oct_gcc
0	-1.70	-1.31	1.10	1.16	-0.57	0.07
1	-1.70	-1.31	0.03	-0.35	-1.12	0.26
2	-1.70	-0.06	-1.62	1.17	-0.53	-0.72
3	-1.70	-0.40	-1.00	-0.03	-1.25	0.06
4	-1.70	-1.65	-0.11	0.76	-1.16	0.10
...	...	...	...	...	...	...
9995	1.71	0.63	1.52	1.50	-1.41	-1.35
9996	1.71	0.77	-1.11	-1.43	0.83	0.23
9997	1.71	0.77	-0.00	-0.68	-0.48	-0.08
9998	1.71	1.60	-1.62	1.26	0.01	1.42
9999	1.71	0.63	0.14	-1.31	0.32	-1.60

10000 rows × 6 columns

Concat back to categorical data

In [76]:
```
1 data = pd.concat([df_num_norm, df[cat_col], df[encoded_col]], axis=1)
```

In [77]:
```
1 data
```

Out[77]:

	Age	Cup-to-Disc Ratio (CDR)	oct_retinal	oct_macular	oct_rnfl	oct_gcc	Gender	Intraocular Pressure (IOP)	Pachymetry	Family History	...	vftr_sensitivity	vftr_specificity	vs_Blurred vision	vs_Eye pain	vs_Halos around lights	vs_Nausea	vs_Redness in the eye	vs_Tunnel vision	vs_Vision loss	vs_Vomiting
0	-1.7	-1.31	1.1	1.16	-0.57	0.07	FALSE	Low	low_median	TRUE	...	0.54	0.72	0	0	1	0	0	0	1	0
1	-1.7	-1.31	0.03	-0.35	-1.12	0.26	FALSE	Low	low_median	TRUE	...	0.91	0.82	0	0	1	0	1	0	1	0
2	-1.7	-0.06	-1.62	1.17	-0.53	-0.72	FALSE	Medium-Low	low_median	TRUE	...	0.95	0.93	0	0	0	0	0	1	1	0
3	-1.7	-0.4	-1	-0.03	-1.25	0.06	FALSE	Medium-Low	low_median	FALSE	...	0.79	0.72	1	0	0	0	1	0	0	1
4	-1.7	-1.65	-0.11	0.76	-1.16	0.1	FALSE	Low	low_median	TRUE	...	0.53	0.77	0	0	0	0	1	1	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	1.71	0.63	1.52	1.5	-1.41	-1.35	FALSE	Medium-High	high_median	FALSE	...	0.92	0.97	0	1	0	0	0	0	1	0
9996	1.71	0.77	-1.11	-1.43	0.83	0.23	FALSE	Medium-High	high_median	FALSE	...	0.89	0.94	0	1	1	0	0	0	1	0
9997	1.71	0.77	0	-0.68	-0.48	-0.08	TRUE	Medium-High	high_median	TRUE	...	0.81	0.76	0	0	1	1	0	0	1	0
9998	1.71	1.6	-1.62	1.26	0.01	1.42	FALSE	Medium-High	high_median	FALSE	...	0.73	0.89	0	0	1	0	0	0	0	1
9999	1.71	0.63	0.14	-1.31	0.32	-1.6	FALSE	Medium-High	high_median	TRUE	...	0.88	1	0	0	0	1	1	0	0	1

FEATURE ENCODING

To convert a dataset into a machine-readable format, all the tagged features in the dataset are converted to numeric format. This helps machine learning algorithms to determine and design better operations on these labeled features.

- One-Hot Encoding
  This technique allows encoding of categorical data by making a separate column.

`Visual Acuity Measurements` column

In [20]:
```
1 df['Visual Acuity Measurements'].value_counts()
```

Out[20]:
```
LogMAR 0.0    2551
LogMAR 0.1    2518
20/20         2489
20/40         2442
Name: Visual Acuity Measurements, dtype: int64
```

We can one-hot encode the `Visual Acuity Measurements` column in the DataFrame. We can use the `get_dummies` function in `pandas`.

This will create new columns for each unique value (i.e. `LogMAR 0.0`, `LogMAR 0.1`, `20/20`, and `20/40`) in the `Visual Acuity Measurements` column and assign binary values (`0` or `1`) based on the presence of each value for each row.

We can rename the newly created columns and add the prefix `vam_` using the `add_prefix` function in `pandas`.

This will create new columns with names like `vam_LogMAR 0.0`, `vam_LogMAR 0.1`, `vam_20/20`, and `vam_20/40`.

In [21]:
```
1  onehotencoded_vam = pd.get_dummies(df['Visual Acuity Measurements'])
2
3  # Add 'vam_' prefix to the new column names
4  onehotencoded_vam = onehotencoded_vam.add_prefix('vam_')
5
6  df = pd.concat([df, onehotencoded_vam], axis=1)
7
8  df = df.drop('Visual Acuity Measurements', axis=1)
9
10 df.head()
```

	Age	Gender	Intraocular Pressure (IOP)	Cup-to-Disc Ratio (CDR)	Family History	Medical History	Medication Usage	Visual Field Test Results	Optical Coherence Tomography (OCT) Results	Pachymetry	Cataract Status	Angle Closure Status	Visual Symptoms	Diagnosis	Glaucoma Type	vam_20/20	vam_20/40	vam_LogMAR 0.0	vam_LogMAR 0.1
0	18	FALSE	11.69	0.36	Yes	Diabetes	Amoxicillin, Aspirin, Ibuprofen, Atorvastatin,...	Sensitivity: 0.54, Specificity: 0.72	RNFL Thickness: 83.35 µm, GCC Thickness: 62.80...	500.99	Absent	Closed	Halos around lights, Halos around lights, Visi...	No Glaucoma	No Glaucoma	1	0	0	0
1	18	FALSE	11.71	0.36	Yes	Diabetes	Amoxicillin, Atorvastatin, Metformin	Sensitivity: 0.91, Specificity: 0.82	RNFL Thickness: 79.39 µm, GCC Thickness: 63.65...	500.81	Absent	Closed	Vision loss, Halos around lights, Redness in t...	No Glaucoma	No Glaucoma	0	0	1	0
2	18	FALSE	17.28	0.54	Yes	Diabetes	Metformin, Ibuprofen, Aspirin, Omeprazole	Sensitivity: 0.95, Specificity: 0.93	RNFL Thickness: 83.64 µm, GCC Thickness: 59.38...	500.79	Absent	Open	Vision loss, Tunnel vision, Tunnel vision	No Glaucoma	No Glaucoma	0	0	0	1
3	18	FALSE	15.79	0.49	No	Diabetes	Ibuprofen, Lisinopril, Atorvastatin	Sensitivity: 0.79, Specificity: 0.72	RNFL Thickness: 78.43 µm, GCC Thickness: 62.77...	500.77	Present	Closed	Blurred vision, Redness in the eye, Vomiting	No Glaucoma	No Glaucoma	0	1	0	0
4	18	FALSE	10.31	0.31	Yes	Diabetes	Atorvastatin, Aspirin	Sensitivity: 0.53, Specificity: 0.77	RNFL Thickness: 79.11 µm, GCC Thickness:	500.77	Present	Closed	Redness in the eye, Vision loss, Tunnel vision	No Glaucoma	No Glaucoma	0	0	1	0

TRAIN-TEST SPLIT

The train and test partition divides the data set in the ratio of 70:30. It is used as a technique to estimate the performance of machine learning algorithms, provide training datasets, and compare the machine's resulting datasets with its learning model to ensure accurate results.

E.4 70:30 SPLIT

In [87]:
```
y = data['y'].values
X = data.drop(columns = ['y'])
```

In [88]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101)
```

In [89]:
```
X_train
```

Out[89]:

	Age	Cup-to-Disc Ratio (CDR)	oct_retinal	oct_macular	oct_rnfl	oct_gcc	Gender	Intraocular Pressure (IOP)	Pachymetry	Family History	...	vftr_sensitivity	vftr_specificity	vs_Blurred vision	vs_Eye pain	vs_Halos around lights	vs_Nausea	vs_Redness in the eye	vs_Tunnel vision	vs_Vision loss	vs_Vomiting
803	-1.46	-1.37	1.58	1.71	1.07	0.91	TRUE	Low	low	FALSE	...	0.66	0.79	1	0	1	0	1	0	0	0
1387	-1.27	-0.06	-0.62	0.84	-0.15	-1.45	FALSE	Medium-Low	low_median	FALSE	...	0.52	0.88	1	0	0	0	0	0	1	1
921	-1.42	-0.06	-1.18	1.39	0.33	-1.38	FALSE	Medium-Low	low_median	FALSE	...	0.8	0.77	0	0	1	0	0	1	0	0
5917	0.34	1.33	1.62	0.49	-1.47	-1.3	TRUE	Medium-High	high	TRUE	...	0.86	0.89	0	0	1	0	1	0	0	1
9610	1.62	0.01	1.58	0.05	-1.56	-1.39	TRUE	Medium-High	high	FALSE	...	0.9	0.87	0	1	1	0	1	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
599	-1.51	-0.75	-0.14	1.69	-0.34	0.49	TRUE	Medium-Low	low	FALSE	...	0.66	0.72	0	0	1	0	0	1	0	0
5695	0.24	0.63	-0.83	1.01	-0.06	-1.54	TRUE	Medium-High	high_median	TRUE	...	0.75	0.89	0	0	1	0	1	0	0	1
8006	1.05	0.63	-1.73	-0.48	0.03	-0.41	TRUE	Medium-High	high_median	TRUE	...	0.86	0.86	1	0	1	0	0	0	0	1
1361	-1.27	-0.2	1.45	1.54	-0.93	1.08	FALSE	Medium-Low	low	TRUE	...	0.7	0.78	0	0	1	0	1	0	0	1
1547	-1.18	-0.68	-0.52	0.28	0.2	-0.22	TRUE	Medium-Low	low	TRUE	...	0.85	0.91	1	0	1	0	1	0	0	0

DASH

https://jwdahmed.pythonanywhere.com/

Previously, it was considered difficult even for developers to create an analytical web-based application for better visualization of data as it required diverse knowledge of various programming languages and frameworks.

Deploying a dash application has made it easier for developers and learners to showcase their analytical results through visualization via an interactive web based application.

Important libraries are imported to provide a foundation for your application. This includes dash, dash core components, dash html components and pandas.

Dash plays the role of initializing the application and dcc allows creation of various interactive components like graphs, drop downs and date ranges etc. Dash html allows user to access all the html tags and lastly panda is required for data comprehension and organising.

Through dash, the layout of the application can be defined using dash html components. Html.div divides the layout of the page into parent and children component. Html.h and html.p are used for headings and paragraphing in the children body respectively. All the main content of the web application including texts, graphs etc are printed in this part.

After defining the layout of the application, flask and dash commands mentioned below are

used. if _name__== "_main_":

app.run_server(debug=True)

This command allows your Dash application to execute locally using the built in server. Debug=true allows user to make modifications in the app which shall be visible in the web app without having to restart the server. Enabling this hot-reloading option makes it user-friendly and easy to manage.

UNIVARIATE ANALYSIS:

https://jwdahmed.pythonanywhere.com/univariate

GLAUCOMA DETECTON

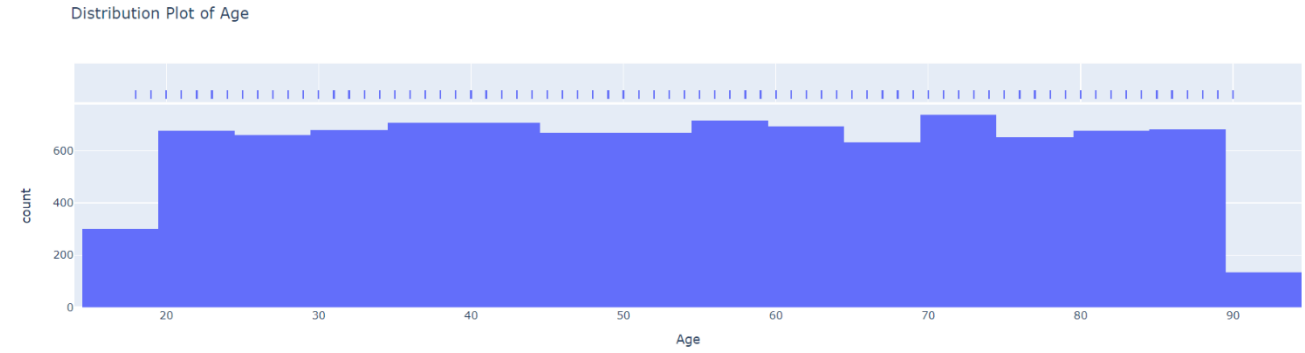
Exploratory Data Analysis

Univariate Analysis

Bivariate Analysis

Trivariate Analysis

Univariate Analysis



BIVARIATE ANALYSIS

https://jwdahmed.pythonanywhere.com/bivariate

GLAUCOMA DETECTON

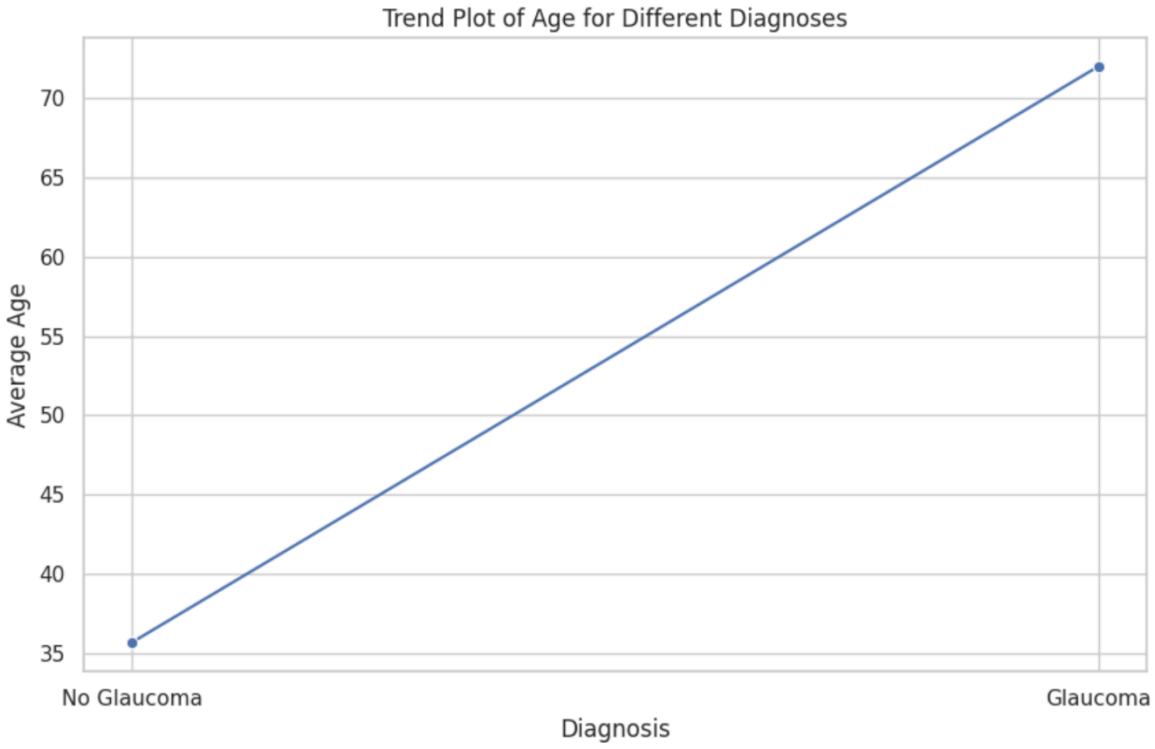
Exploratory Data Analysis

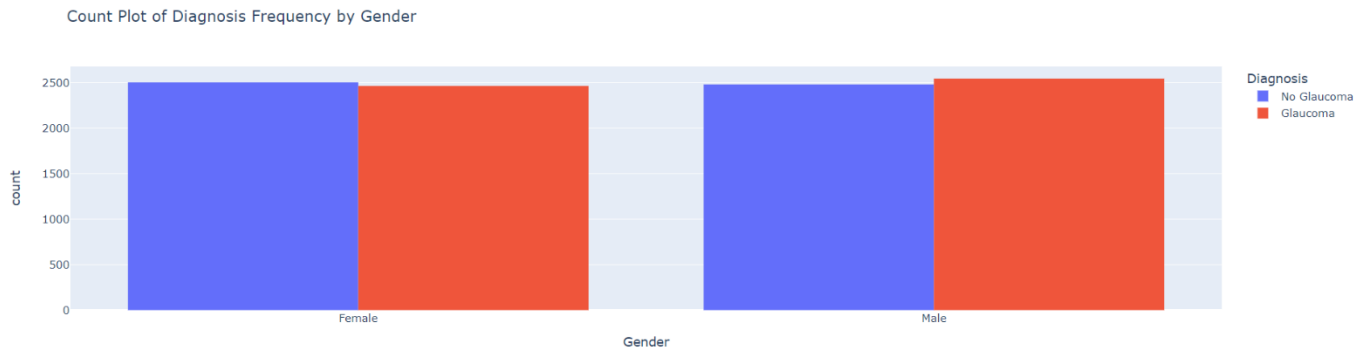
Univariate Analysis

Bivariate Analysis

Trivariate Analysis

Bivariate Analysis





Trivariate Analysis

https://jwdahmed.pythonanywhere.com/trivariate

GLAUCOMA DETECTON

Exploratory Data Analysis

Univariate Analysis

Bivariate Analysis

Trivariate Analysis

Trivariate Analysis