**Abstract:**

The rising prevalence of chronic diseases like **Chronic Kidney Disease** (CKD) and **Parkinson's Disease** underscores the need for early diagnosis and intervention to improve patient outcomes. This project aims to develop a multi-disease prediction system leveraging machine learning models to accurately predict the likelihood of CKD and Parkinson's Disease based on medical data. The system integrates two specialized predictive models, each trained on distinct datasets corresponding to CKD and Parkinson's Disease. Features such as age, blood pressure, specific gravity, albumin, sugar levels, red blood cells, and additional biomarkers are used for CKD prediction. For Parkinson's Disease,relevant clinical data are incorporated.

The models are developed using state-of-the-art machine learning techniques to ensure high accuracy and reliability. The system is deployed using Flutter, providing a cross-platform, user-friendly interface that allows healthcare professionals and users to access the predictive models seamlessly on both Android and iOS devices. This deployment strategy ensures that the system is easily accessible, facilitating early detection and management of these diseases.

**Pakrinson’s Disease**

**1.Pandas (import pandas as pd)**

* **Purpose**: Pandas is a powerful library for data manipulation and analysis.
* **Usage**: In the context of Parkinson's disease, Pandas is used to load and manipulate the dataset containing features relevant to the disease, such as motor symptoms, voice measurements, and other clinical data. This dataset is then structured into a DataFrame, allowing you to efficiently explore, clean, and prepare the data for model training.

**2. NumPy (import numpy as np)**

* **Purpose**: NumPy is the fundamental package for numerical computation in Python.
* **Usage**: For Parkinson's disease prediction, NumPy helps in performing numerical operations on the dataset, such as converting data into arrays, handling missing values, and performing mathematical operations that are necessary for feature engineering. It ensures that the data is in a format suitable for machine learning algorithms.

**3. Scikit-learn (from sklearn.model\_selection import train\_test\_split)**

* **Purpose**: Scikit-learn is a widely used machine learning library that provides tools for model training, evaluation, and data preprocessing.
* **Usage**:
  + train\_test\_split: In the Parkinson's disease project, this function is used to split the dataset into training and testing sets. This ensures that the model is trained on one portion of the data and tested on another, allowing you to evaluate how well the model generalizes to unseen data, which is crucial for reliable disease prediction.

**4. StandardScaler (from sklearn.preprocessing import StandardScaler)**

* **Purpose**: StandardScaler is a tool for feature scaling, which is an essential preprocessing step in machine learning.
* **Usage**: For the Parkinson's disease model, StandardScaler is used to standardize the features such as voice measurements, gait analysis data, and other clinical indicators by removing the mean and scaling to unit variance. This step is vital because it ensures that all features contribute equally to the model, preventing those with larger numerical values from disproportionately influencing the model's predictions.

**5. Classification Report & Precision Score (from sklearn.metrics import classification\_report, precision\_score)**

* **Purpose**: These are performance metrics provided by Scikit-learn.
* **Usage**:
  + classification\_report: This tool provides a comprehensive breakdown of the Parkinson's disease model's performance, including metrics like precision, recall, and F1-score for each class (e.g., whether a patient has Parkinson's or not). This helps in understanding the strengths and weaknesses of the model.
  + precision\_score: Specifically, in the context of Parkinson's disease prediction, precision is critical. A high precision score indicates that the model makes few false positive errors, which is crucial in medical diagnostics to avoid incorrectly identifying healthy individuals as having Parkinson's.

**6. XGBoost (from xgboost import XGBClassifier)**

* **Purpose**: XGBoost is an optimized gradient boosting library designed for speed and performance.
* **Usage**: XGBClassifier is employed to build a robust classification model that predicts whether a patient has Parkinson's disease based on the input features. XGBoost is particularly well-suited for this task due to its ability to handle complex patterns in the data, which is essential given the subtle and varied symptoms of Parkinson's disease.

**7. Joblib (import joblib)**

* **Purpose**: Joblib is a library for efficient serialization of Python objects, especially for large NumPy arrays.
* **Usage**: In your Parkinson's disease prediction project, Joblib is used to save the trained XGBoost model to disk. This allows the model to be loaded and used for predictions without needing to retrain it, which is crucial for deployment in a real-world medical application where timely predictions are necessary.

**8. Matplotlib (import matplotlib.pyplot as plt)**

* **Purpose**: Matplotlib is a plotting library for creating static, animated, and interactive visualizations in Python.
* **Usage**: Matplotlib is used to create visualizations that help interpret the Parkinson's disease model’s results. For example, you might use it to plot the feature importance, ROC curves, or confusion matrices, which can provide insights into how well the model is performing and which features are most influential in predicting Parkinson's disease.

**9. Seaborn (import seaborn as sns)**

* **Purpose**: Seaborn is a Python visualization library based on Matplotlib that provides a high-level interface for drawing attractive and informative statistical graphics.
* **Usage**: Seaborn is used to create more sophisticated and visually appealing plots that help analyze the Parkinson's disease dataset. For instance, you might use it to plot correlations between different symptoms and the presence of Parkinson's, or to visualize the distribution of predictions made by the model. These visualizations can aid in better understanding the data and the model’s performance.

**Summary:**

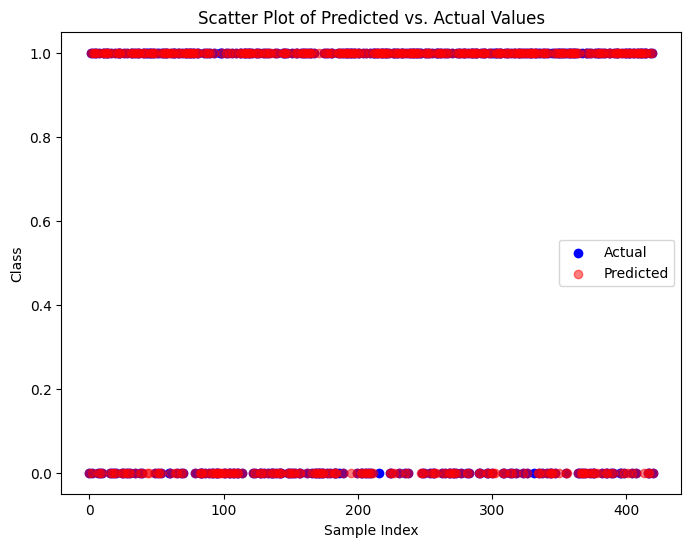
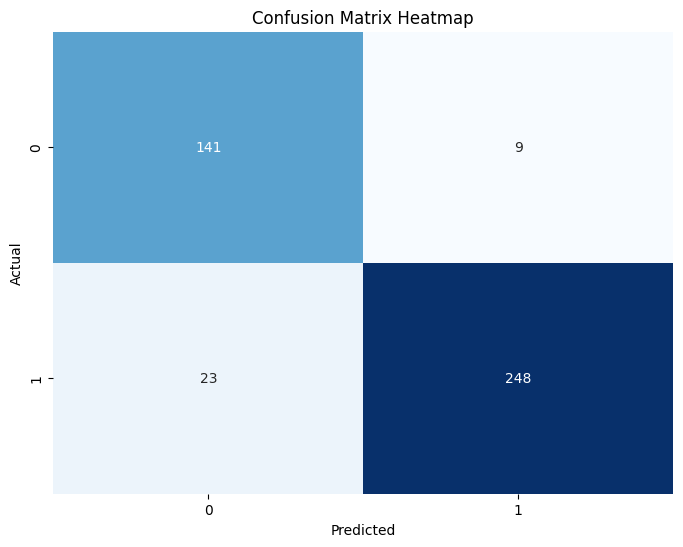
Each of these libraries is critical in different stages of your Parkinson's disease prediction project, from data preprocessing (Pandas, NumPy, StandardScaler) to model training and evaluation (Scikit-learn, XGBoost), and finally to model persistence (Joblib) and visualization (Matplotlib, Seaborn). They collectively enable the development of a robust, accurate, and interpretable machine-learning model tailored for predicting Parkinson's disease based on clinical data.

**Visualization:**

Top of Form

Bottom of Form





**Accuracy:**

Training Accuracy of XgBoost is 1.0

Test Accuracy of XgBoost is 0.9239904988123515

Confusion Matrix :-

[[141 9]

[ 23 248]]

Classification Report :-

precision recall f1-score support

0 0.86 0.94 0.90 150

1 0.96 0.92 0.94 271

accuracy 0.92 421

macro avg 0.91 0.93 0.92 421

weighted avg 0.93 0.92 0.92 421

**Kidney Chronic Disease:**

* 1. **Handling Missing Values**
* Random Value Imputation: For features with a significant amount of missing values, such as red\_blood\_cells and pus\_cell, a random value imputation technique is employed. This method replaces missing values with randomly selected samples from the available data in the same column, maintaining the original data distribution.
* Mode Imputation: For categorical columns with fewer missing values, the mode (most frequently occurring value) is used to fill in the gaps. This approach ensures that missing data points are filled with the most representative value from the existing data.

**2. Label Encoding**

* Purpose: Categorical data is converted into numerical form using Label Encoding. This process assigns each unique category within a feature an integer value, enabling the machine learning model to process categorical information effectively.

**3. Data Splitting**

* Train-Test Split: The dataset is divided into training and testing sets. The training set, comprising 70% of the data, is used to train the model, while the remaining 30% is reserved for testing the model's performance. This split ensures that the model is evaluated on unseen data, providing a realistic measure of its predictive power.

**4. Feature Scaling**

* Standardization: Feature scaling is performed using StandardScaler, which standardizes features by removing the mean and scaling to unit variance. This step is essential for ensuring that features with different scales, such as age, blood pressure, and other medical measurements, contribute equally to the model's learning process.

**5. Model Training with XGBoost**

* XGBoost Classifier: The XGBoost algorithm is used to train the model, specifically configured for binary classification. XGBoost is chosen for its efficiency and accuracy in handling large datasets with complex relationships. The model is trained using specific parameters, such as learning rate, maximum tree depth, and the number of trees, which are fine-tuned to achieve optimal performance.

**6. Model Evaluation**

* Accuracy Score: The accuracy score is calculated to measure the proportion of correct predictions made by the model on both the training and test datasets.
* Confusion Matrix: A confusion matrix is generated to visualize the model's performance, showing the number of true positives, false positives, true negatives, and false negatives. This matrix helps identify how well the model is distinguishing between CKD and non-CKD cases.
* Classification Report: The classification report provides detailed metrics, including precision, recall, and F1-score for each class. These metrics are crucial for evaluating the model's effectiveness in predicting CKD, particularly in medical contexts where false positives and false negatives have significant implications.

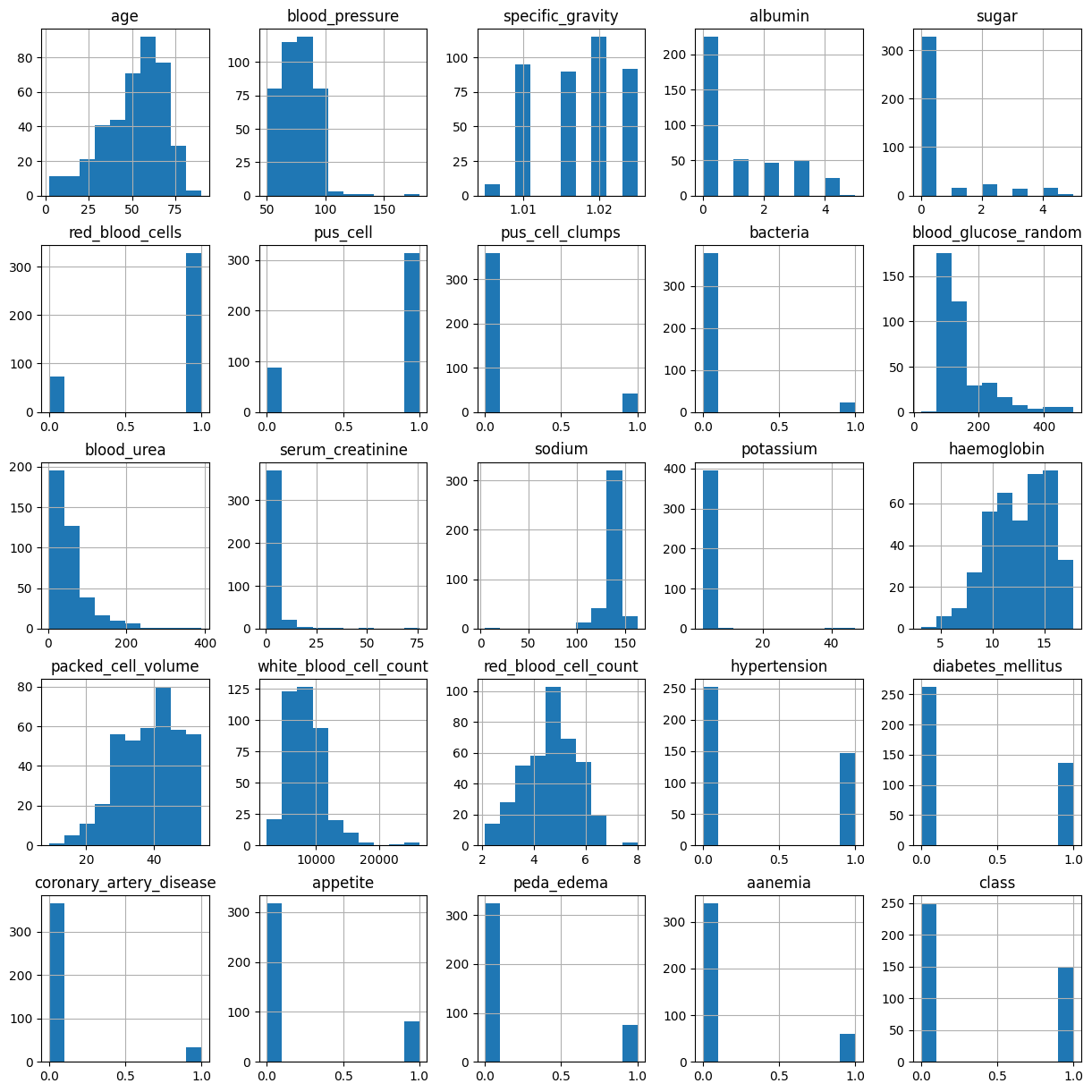
**7. Making Predictions**

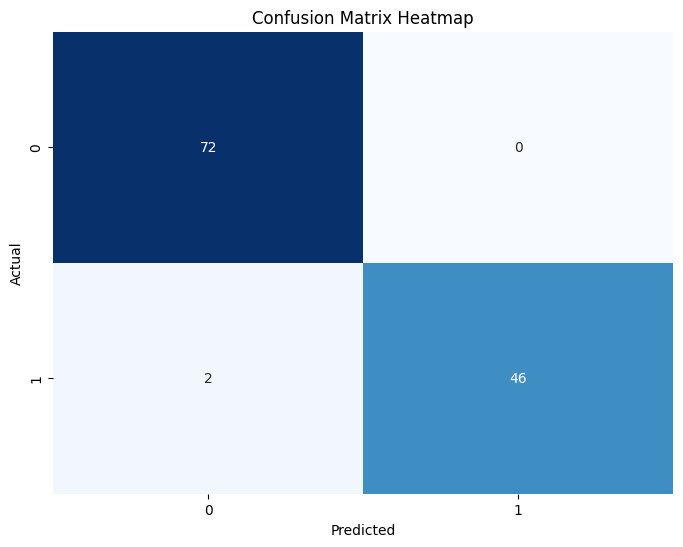
* Prediction on New Data: The trained model is used to make predictions on new, unseen data. This step involves inputting patient-specific medical features into the model, which then outputs a prediction indicating whether the patient has CKD.

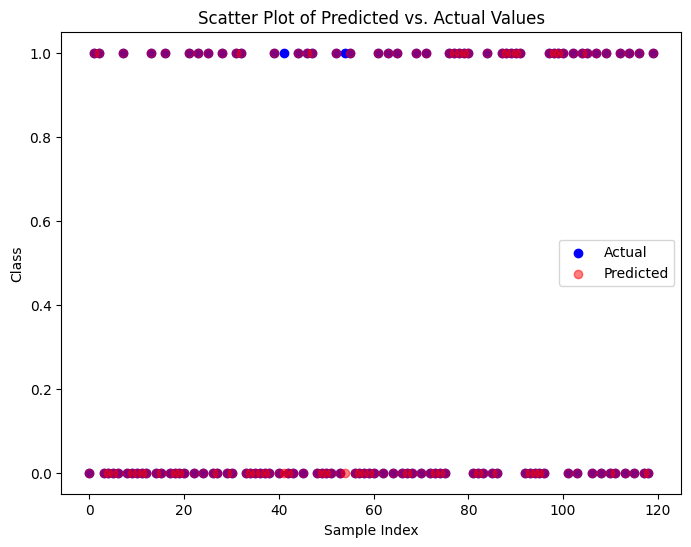
**Summary:**

Developing a machine learning model for predicting Chronic Kidney Disease. Each step, from data preprocessing and feature scaling to model training and evaluation, is carefully executed to ensure that the model is accurate and reliable for real-world medical applications. The use of XGBoost, combined with techniques like imputation and standardization, helps create a robust model capable of effectively identifying CKD in patients based on their medical data.

**Visualization:**







**Accuracy:**

Training Accuracy of XgBoost is 1.0

Test Accuracy of XgBoost is 0.9833333333333333

Confusion Matrix :-

[[72 0]

[ 2 46]]

Classification Report :-

precision recall f1-score support

0 0.97 1.00 0.99 72

1 1.00 0.96 0.98 48

accuracy 0.98 120

macro avg 0.99 0.98 0.98 120

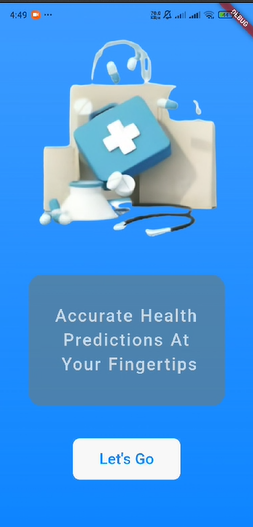
weighted avg 0.98 0.98 0.98 120  
  
  
  
  
**Abstract:**

The app we 've created is built with Flutter and Dart, focusing on basic functions to keep things simple and user-friendly. The frontend is easy to navigate, thanks to Flutter's design capabilities.

For the backend, I've integrated a machine learning model, which the app accesses through an API. This allows the app to perform smarter tasks, adding more value beyond just basic features.

Overall, this project shows my growing skills in both app development and machine learning, combining them to create a functional and innovative app.

**Front-End:**

****

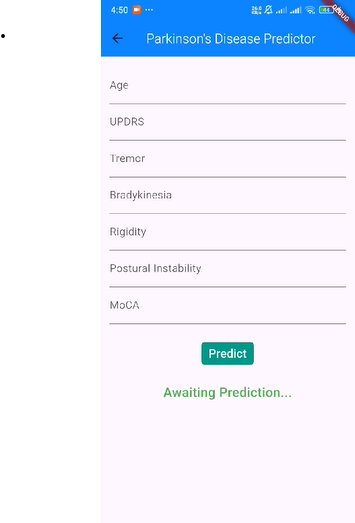
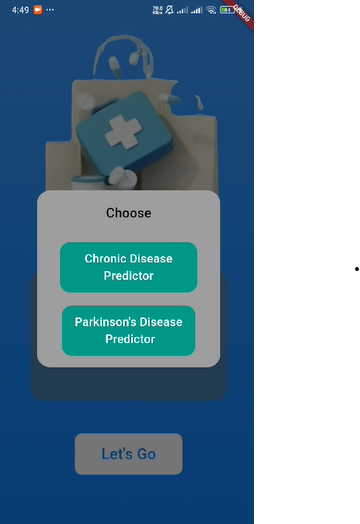
The app uses a specific Flutter/Dart library for animations, creating dynamic visual effects. It's also capable of much more, offering enhanced functionality beyond just basic animations.

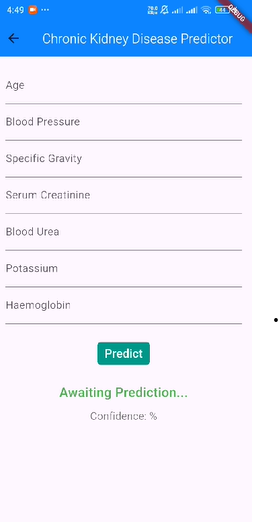
**Animated-Text(animated\_text\_kit:):**

The Animated Text Kit in Flutter/Dart is a helpful library that makes it easy to add animated text effects to your app. It includes different types of animations, like typewriter, rotating, fading, and scaling text, which make your app look more lively and engaging. You can customize these animations by changing the speed, colors, and styles to fit your app’s design.The library is easy to use, even for beginners in Flutter. Adding these text animations can improve the user experience, making your app more visually appealing and interactive with little effort.  
The text (“**Accurate Health Prediction at Your FingerTips**”) written in blue container is animated.

**Functionalty:**

As just pressing the button the following box appears.





The following pages are designed to appear as needed, helping to enhance the accuracy of disease prediction. They assess how likely a person is to have the disease and provide guidance on when to begin medication to prevent serious harm. By measuring the prediction's effectiveness, these pages offer a clearer understanding of the potential risk, allowing for timely intervention. This approach ensures that individuals can take necessary action early, improving outcomes and helping to manage   
the disease before it becomes a more severe health issue

**Result**:

Our app integrates a machine learning model via API to predict disease risk and guide timely medication.