AI HEALTH GUARD

Your Personalized Health Advisor. Predicts diseases, offers tailored medical advice, workouts, and diet plans for holistic well-being.

Capstone-I project report submitted

by the student of

Hybrid UG program in Computer Science & Data Analytics

Alok Kumar Choudhary
Roll No.- 2312RES77

INDIAN INSTITUTE OF TECHNOLOGY PATNA
BIHTA - 801106, INDIA

Date - 25 June 2024

Declaration

I hereby declare that this submission is my own work and that, to the best of our knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Date: 25 June 2024

Student Name - Alok Kumar Choudhary
Roll No - 2312RES77
Group No - 8
Signature - Alok Kumar Choudhary

Summary of the Project

The AI Health Guard Project has been envisioned to bring a revolution in healthcare accessibility to people in remote and rural areas, through the most advanced techniques in Data Science and Machine Learning. The project focuses on early diagnosis, personalized health counseling, and preventative advice in these regions facing a shortage of medical professionals. It uses a large dataset from sites like Kaggle and involves the construction of a complete workflow, starting from data collection and preparation to model training and validation.

In Phase-1, we developed a prediction system for diagnosis and provision of related health tips based on symptoms reported by a user. The model developed uses a Support Vector Classifier (SVC) model that has been able to achieve determined high accuracy in predicting illnesses in a user and giving advice tailored to user medication, diet, workouts, and precautions.

In Phase-2, the scope is expanded to include predictive models for pregnancy-related risk, heart diseases and diabetic disease prediction using Random Forest and Logistic Regression algorithms respectively, both optimized to offer high accuracy.

So, in terms of the key functionalities of the project, it involves user-friendly interfaces, data preprocessing, feature engineering, model training, hyperparameter tuning, and web application being developed so that these models can have a very broad use with Python, Flask, and JavaScript.

This project highlights the potential that AI has for bridging healthcare gaps and making quality healthcare advice accessible to every corner.

******

Contents

1. Introduction

In remote and rural areas with few medical professionals, healthcare becomes a serious problem. The AI Health Guard Project is a new beacon for such regions. Applying Data Science and Machine Learning (ML) technologies, it seeks to link the far reaches of health services with neglected population groups.

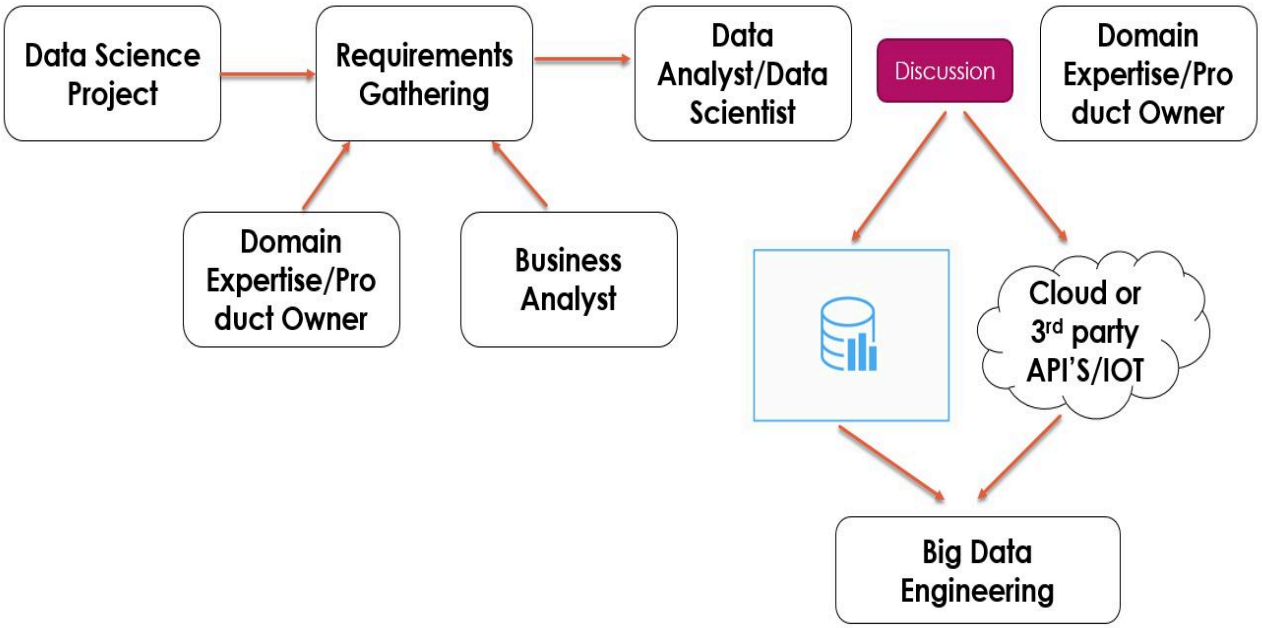
The AI Health Guard project is a new frontier in healthcare provision. Its focus is on quick diagnosis, customized health counseling and preventative advice in a part of the world where there are few doctors to turn to. With the superior algorithms and creative platform technologies of the project, it's hoped to enable individuals to achieve wellness proactively rather than passively.
Specifically, the AI Health Guard project's design element is to address the needs of rural communities, where healthcare professionals' shortage typically results in late diagnosis and treatment. This project, being built upon the capabilities of data science and ML, aims to deliver a solution that not only helps identify the disease but also make actionable suggestions and recommendations based on the user's data.

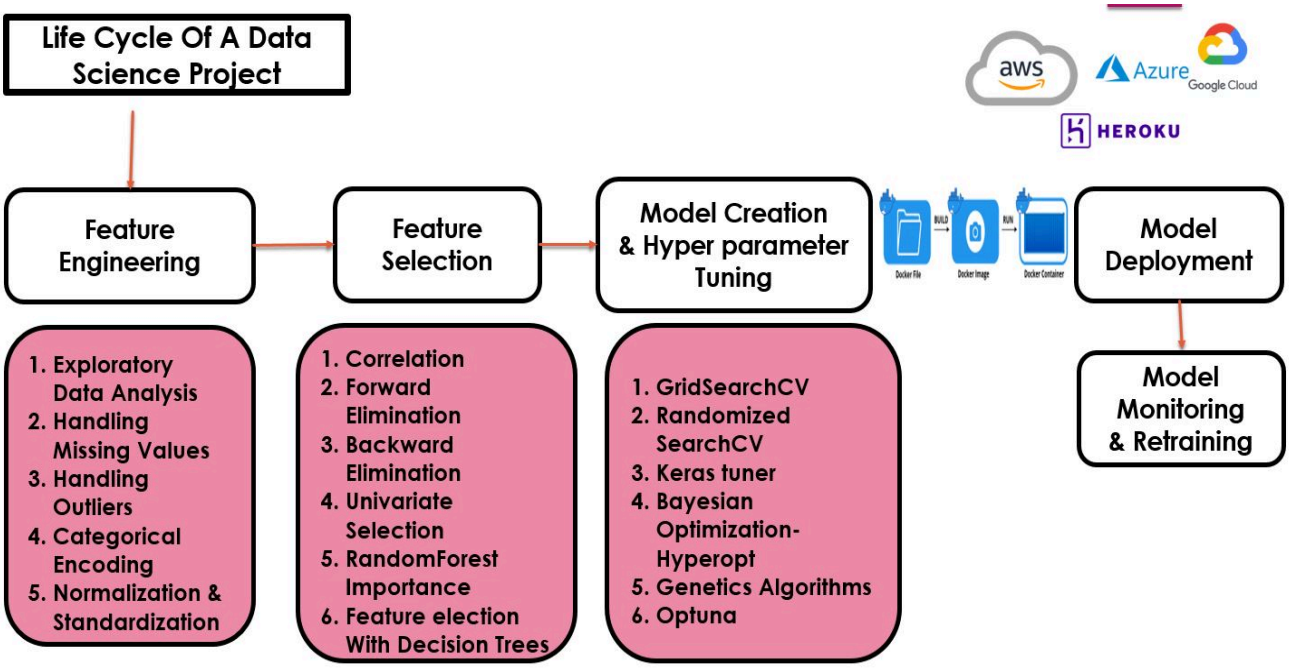
The primary purpose of the AI Health Guard project is to deliver users a trustworthy and available platform for health status diagnosis, accurate disease identification, and personalized recommendation for maintaining and improving the state of health. The datasets the project will be using for its operations will be taken from reliable platforms, such as Kaggle, and databases from API, to ensure the level of information accuracy and relevancy.

The workflow of the project involves a number of stages such as data acquisition, preprocessing, model training, and result evaluation. Based on detailed data analysis and precise feature engineering, the project attempts to secure as a result meaningful outcome from raw datasets substantially affecting the performance and reliability of the predictive models.
Additionally, the AI Health Guard project pays special attention to the user interface creating a web application developed with Python, Flask, and JavaScript,etc... Hence, end-users have the opportunity to easily interact with the complex including input of symptoms detection and receiving personalized recommendations contributing to the greater availability and user orientation of healthcare.
Overall, AI Health Guard is set to be a groundbreaking addition to the field of health technology, offering a proactive and personalized approach to health maintenance and disease prevention.

2. Process and Workflow

I learned how data science & machine learning projects work and what process follows in these projects, so here I put my understanding of these industry level projects in the form of images.





******

3. Learning Phase

The learning phase of my project, so different techniques and technologies are being tried and tested. Also, the project uses the Python programming language for tasks related to Data Science and machine learning. Data collection is dependent on available datasets from platforms such as Kaggle. The system is designed to include features like symptom input, data analysis, and recommendation generation. It will provide results of medication and diet, workout advice, and things to be careful about based on symptoms.

Solution Approach:

The system has the ability to analyze user input symptoms with the help of data science and machine learning in order to provide personalized recommendations. It includes several key techniques of data science and machine learning algorithms, which are listed below.

Data Science

The project was built on a comprehensive foundation of data science techniques, Here I show what I learned during this project:

- **Python:** Python programming language used for data manipulation and model development, because it relies on extensive libraries and has great community support.
- **Feature Engineering:** The transformation of raw data into features that are good representations more closely approximating the underlying problem, bringing better model accuracy.
- **Feature Selection:** The strategy of picking a part of the relevant features for modeling, which helps in reducing models, preventing overfitting, and enhancing generalization ability.
- **Exploratory Data Analysis (EDA):** A very important step to cover the main characteristics of the dataset, usually using visual methods. This is in order to find regularities, exceptions, initiative, and then validate hypotheses.

Statistical Tools

Understanding of statistical tools was critical to analyzing data and making decisions. For example:

- **Sampling:** A technique designed to estimate characteristics of white population by choosing a subset of members for testing . The chosen subset is a sample of the population.
- **Hypothesis Testing :** A statistical method to assess a sample of the population to prove or disprove a hypothesis on the parameter of the population.
- **Power Law Distribution:** A relationship in which a change in one unit results in a proportional change in another unit relative to its initial value . In this case, the size of one population directly impacts another.
- **Type 1 and Type II errors:** Errors related to hypothesis that occur when either valid or invalid statistics are disregarded.
- **Confidence Interval and Margin of Error**: confidence interval and margin of error are statistical tools providing a range of values within which the population parameter is expected to lie with a certain level of confidence.

Database Management

This module has provided me knowledge about the management of data with professional suitability for MySQL:

MySQL: This is a relational database management system that allows data to be stored and accessed by different software applications. Some key points about MySQL are:

- **Database Design:** A database's organization into a collection of tables for storing and retrieving data, laying out the relationship between tables, and specifying the design and kind of data that will be stored. Essentially, it involves creating tables, connecting relationships between the tables, and defining primary and foreign keys.
- **Data Modeling:** Creating efficient data models that reflect the system's requirements and ensure data integrity.
- **SQL Queries:** SQL will be used to interact with the data. It includes writing queries to insert, update, delete, and extract data.
- **Indexing:** A procedure in which indexes on columns are created to accelerate data access.
- **Normalization:** This is a process in which data will be organized to minimize redundancy and avoid data inconsistency. It is a process of arranging data in the database. It includes splitting the tables into smaller ones and defining the relationship between them.

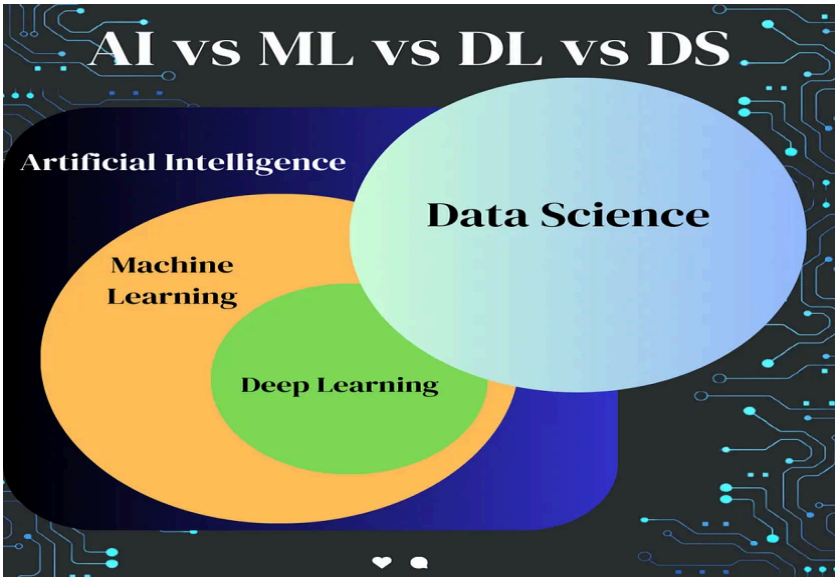
Machine Learning Algorithms

> **“ Here I am only writing an overview of the concept of machine learning and not in detail. Besides, the machine learning algorithm will be executed directly in the project. “ Learning and execution” ”**

Machine Learning forms part of AI's field. It builds a predictive model based on training data. Predictive Power Memorial and Its Architect ML is the soul of AI. It enables how machines learn from data to bold in case of experience. It’s the science of getting algorithms to detect hidden patterns and make them decide without explicit programming.

In this, we will look at the two fundamental types of ML which are

- Supervised learning
- Unsupervised learning



I learned Differences Between AI, ML, DL, DS.

Supervised Learning:

In supervised learning, you have labeled data. You use this data to train the algorithm so that it can learn from examples and predict new instances based on past results. The goal is for a mapping function, f: X → Y

Here are some common forms of supervised learning:

a. Regression

Output from regression models is continuous numeric. For example:

- **Linear Regression-** Linear relationships between input features which are eligible for continuous output.
- **Example:** Based on features like the number of bedrooms or square footage, predict house prices on him/her behalf.

b. Classification:

Classification models input data to predefined classes or categories. For example:

- **Logistic Regression-** Binary classification problems, such as identifying whether an email is spam or not.
- **Random Forests-** Composite collections of decision trees.
- **Decision Trees-** A series of tree like structures that compete with feature properties as conditions.
- **Support Vector Machines (SVM)-** Based on the optimal technology hyperplanes can discover examples of different classes.
- **Naive Bayes-** A group of simple probabilistic classifiers that are based on the idea of applying Bayes' theorem with strong independence field assumPtions between the features.
- **K-Nearest Neighbors (KNN)-** KNN is used to classify and predict a non-parametric process.
- **Gradient Boosting Machines (GBM)-** A model which follows this method is called ensemble (short for XTrain x Random Access).
- **Example:** Email classification.

Unsupervised Learning

In unsupervised learning, algorithms find hidden patterns in unlabeled data. Common methods include:

a. Clustering: Groups similar data points together. Examples:

- **K-Means**: Forms K clusters based on data point similarities.
- **Hierarchical Clustering**: Creates a nested structure of clusters.
- **DBSCAN**: Identifies clusters based on the density of data points.
- **For example,** customer segmentation automatically arranges individuals based on their diverse purchasing behaviors over time.

b. Dimensionality Reduction

Dimensionality reduction techniques condense features while maintaining core aspects, like squeezing a sponge to remove excess water.

- **Principal component analysis** rotates the coordinate system to align with directions of maximum variance, projecting onto new orthogonal axes.
- **t-SNE maps** high-dimensional data to points in a lower-dimensional space, allowing visualization of relationships between single examples or groups in ways unseen before.
- **For example,** face recognition algorithms may preprocess images through dimensionality reduction to distill features for more accurate matching.

Additional Machine Learning Concepts

I learned other critical advanced machine learning concepts including:

Overfitting and Underfitting: Over- and Under-fitting are key in understanding the relationship between the complexity of the model and generalization.

Measuring Performance: Ability to evaluate the performance of the model using performance metrics such as accuracy, precision, recall, F1 score, and ROC-AUC.

Hyperparameter Tuning: Tuning adjustable parameters of the model in order to produce the best performing model.

Ensemble Techniques: Learning how to take different models for improved predictive performance and combine them into one, a process called ensembling. This involves using techniques like bagging and boosting.

Outliers: Detecting and treating outliers within the data to make it more robust.

The project will use integrated techniques in these areas and tools to seek informative and precise recommendations from the user's symptoms by the power of data science and machine learning.

******

4. Symptoms-Disease Prediction

Now, I divide this project into 2 phases. This is a Phase-1 of our project. In this phase I train only the Symptoms-Disease prediction model. After that in phase 2 I trained different models to upgrade our project.

4.1 Data Acquisition

“ The data for "AI HEALTH GUARD" is taken from Kaggle, a platform for data scientists and machine learning engineers. This dataset includes 8 CSV files“

- **Main_Dataset.csv:** The main dataset used for the Symptoms-Disease prediction model.
- **description.csv:** Gives detailed descriptions of the health conditions.
- **diets.csv:** Provides information about which diets are appropriate for various health conditions.
- **workout_df.csv:** Lists planned ways that are suited to an individual's specific health demands and encompasses work-outs combined with lifestyle advice for a healthy lifestyle.
- **medications.csv:** Gives details of when and how to take what kind of medication, should you need some.
- **precautions_df.csv:** Lists the different precautions that you are advised to adopt when facing various health conditions.
- **symtoms_df.csv:** Contains an exhaustive list of symptoms presented by different illnesses.
- **Symptom-severity.csv:** Describes the severity of specific symptoms.

4.2 Data Preprocessing

Data Preprocessing steps were executed to cover the dataset's examination and transformation needs so that missing variables could be catered for, categorical variables encoded, and visualizations created that make it easier to understand its contents. These are critical in enabling the construction of powerful machine learning models as well as understanding how symptoms relate to diseases.

Initial Data Inspection:

- There are 10961 entries and 606 columns in the dataset.
- Each row in the dataset corresponds to a distinct patient's information records, whereby one of the columns represents the illness while the other columns represent different symptoms.
- The dataset had to be reshuffled so that records are randomly distributed, which makes it possible for other machine learning models to perform better and be more reliable.

	Disease	mucoid_sputum	blackheads	mood_swings	movement_stiffness	bladder_discomfort	swelling_joints	yellow_urine	increased_appetite	loss_of_balance	...	itching	dehydration	coma	nausea	irritability	dizziness	headache	vomiting	diarrhoea	anxiety
4984	Tularemia	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4710	Arthritis	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9413	Stroke	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
6587	Lead poisoning	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
10418	Sickle-cell anemia	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2465	Myocardial Infarction (Heart Attack)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
10154	Rabies	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3299	Myocardial Infarction (Heart Attack)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
4726	Warkany syndrome	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6143	Myocardial Infarction (Heart Attack)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

10 rows × 606 columns

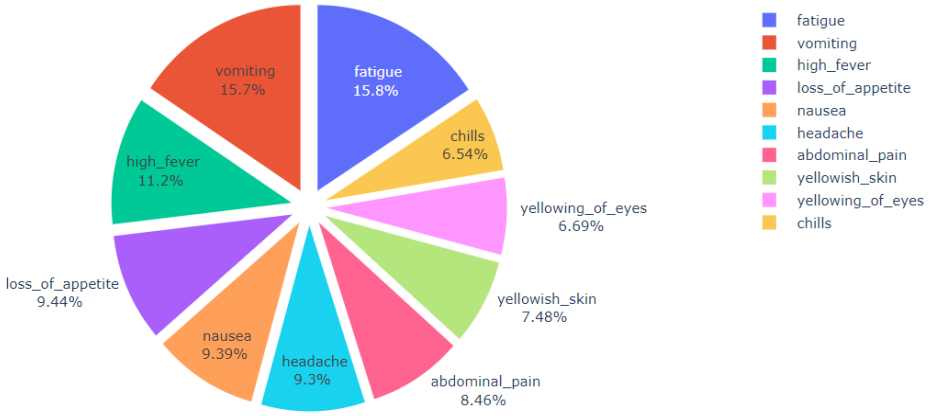
Original Dataset

Data Shape and Types:

- The dataset has a shape of (10961, 606).
- The Dataset contains 287 unique diseases.
- The Dataset contains 606 unique symptoms.
- All columns are of object data type, indicating that they contain categorical data.

Symptom Analysis:

- Frequency of Each Disease plots represent approximately all disease has the same count frequency that is approx 120.
- Top Symptoms in datasets is :-



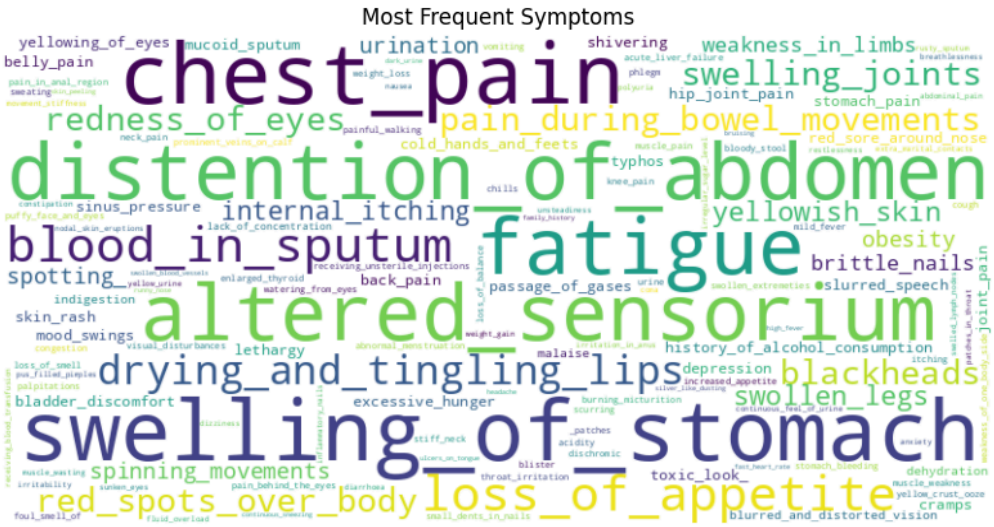
Data Transformation

- **Unique Symptoms Identification**
  - The unique symptoms across all columns were identified, leading to a comprehensive list of symptoms present in the dataset. This step ensured that all symptoms were accounted for in the analysis.
- **Binary Encoding of Symptoms**:
  - These alterations lead to a newly generated matrix, which includes 606 symptom columns (indicating symptoms' presence or absence) and a single disease predictor column ('Disease').
  - Every indicator of disease was translated into either 0 or 1.
- **Column Name Cleaning**:
  - Column names were stripped of any leading or trailing whitespace to ensure consistency.
- **Saving Transformed Data**:
  - The unique column names (symptoms) and the transformed data frame were saved to CSV files for future use.

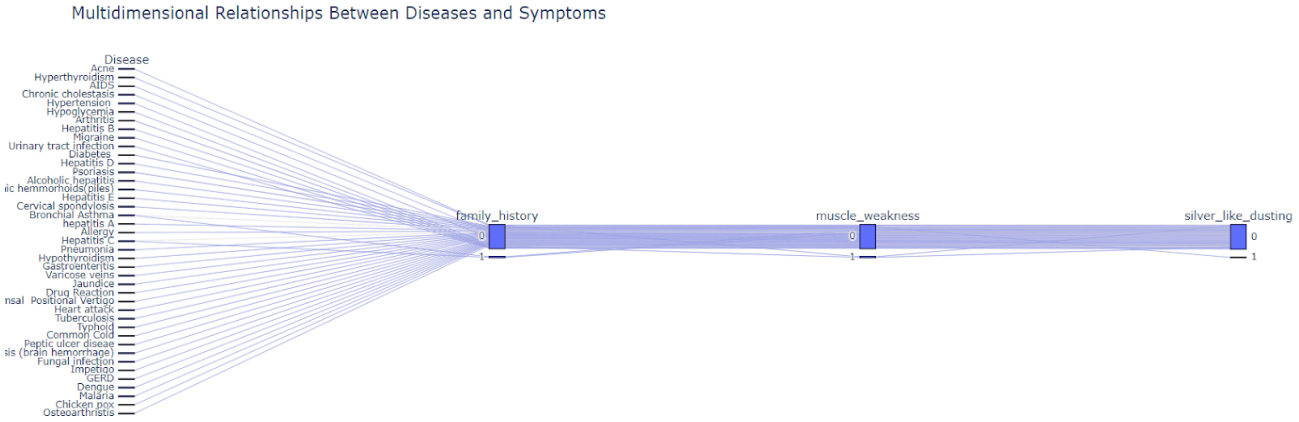
Visualizing Relationships

NOTE :- **The visualization of the data has been done on some samples of the original dataset.**

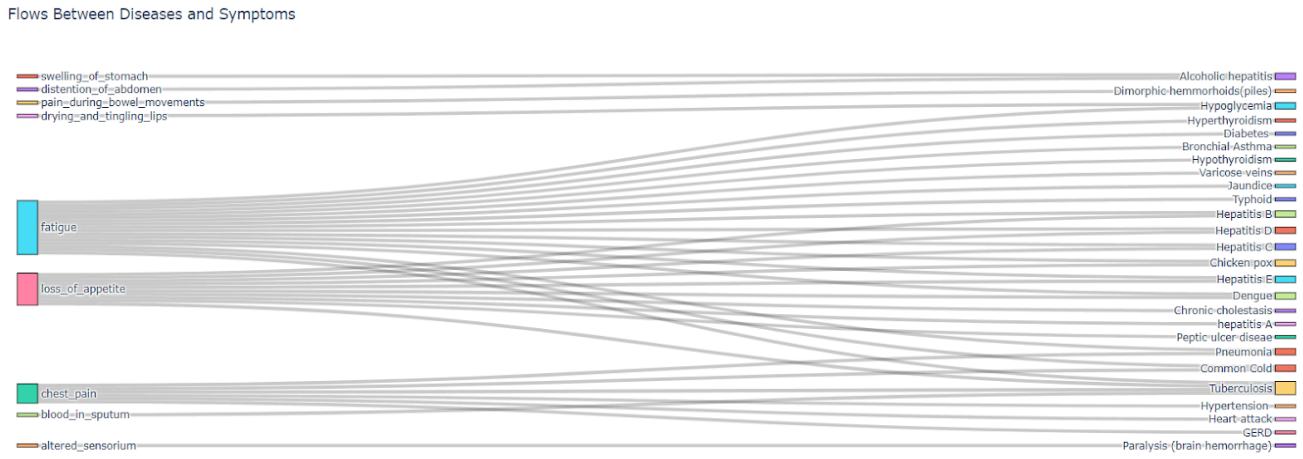
- **Word Cloud of Symptoms:**
  - A word cloud was created to visualize the most common symptoms across the dataset.



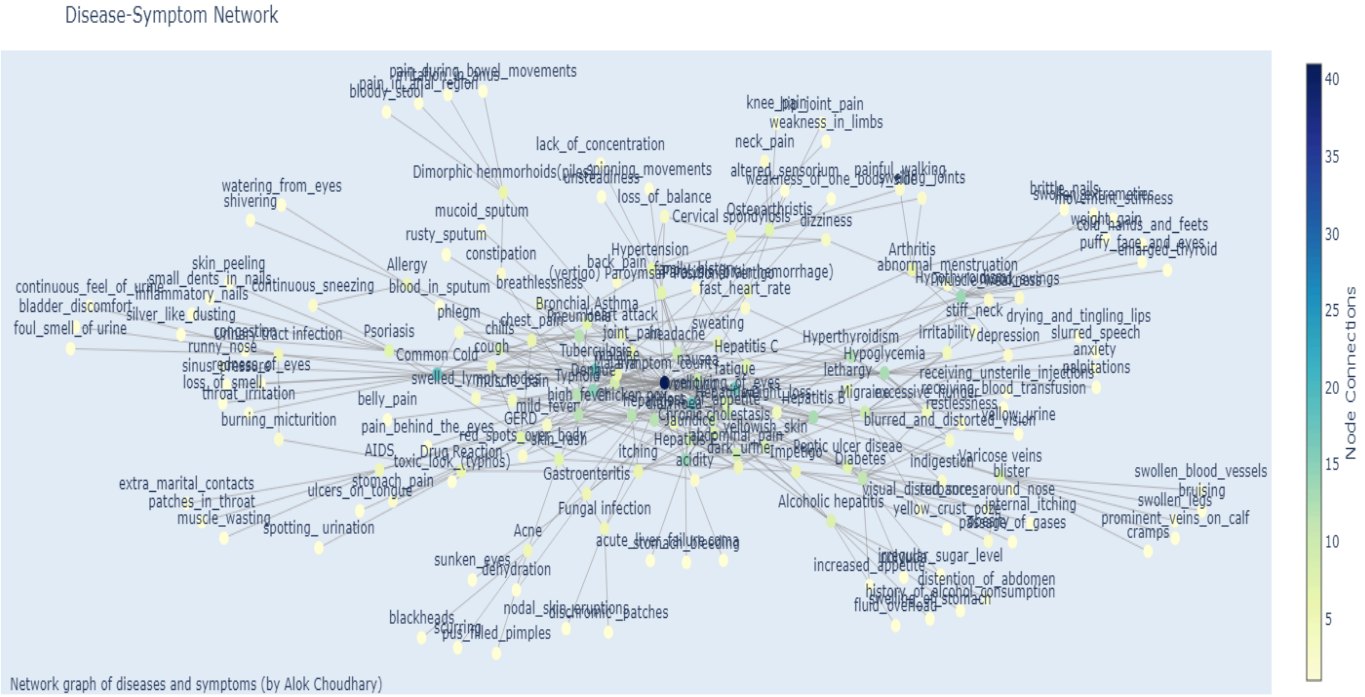
- **Multidimensional Relationships Between Diseases and Symptoms:**
  - It shows relationships between multiple categorical variables. In this example, the categories are different diseases and symptoms. Each vertical axis represents a single variable, and each data point is a combination of categories across all variables. Lines connect the data points, showing how different categories co-occur.



- **Flows Between Diseases and Symptoms:**
  - The diagram shows the flow of patients between different diseases and related symptoms.
  - The width of the arrows represents the number of patients experiencing a particular symptom or disease.



- **Disease-Symptom Network:**
  - Nodes in the graph represent various diseases and symptoms. Edges connect nodes that frequently co-occur together. Thicker edges indicate stronger connections.



"By executing these preprocessing steps, the dataset was thoroughly prepared for further analysis and modeling. These steps ensured the data was clean, well-structured, and suitable for building predictive models."

4.3 Model Training

This section gives a complete description of the ways and principles applied in training the predictive models for diagnosing diseases from their symptoms.

Train-Test Split:

- We divided data into two subsets using an 80-20 split. 8768 samples were included in the training set and 2193 samples were included in the testing set. This particular division provides valuable data for models to learn from as well as a test set that represents the whole dataset.

Shape of the datasets:-

```
print("Shape of X_train : ", X_train.shape)
print("Shape of X_test : ", X_test.shape)
print("Shape of y_train : ", y_train.shape)
print("Shape of y_test : ", y_test.shape)

# Output :-
Shape of X_train :  (8768, 605)
Shape of X_test :  (2193, 605)
Shape of y_train :  (8768,)
Shape of y_test :  (2193,)
```

Model Selection:

- We trained seven different classification models to determine the best algorithm for predicting diseases from symptoms.
- The models were trained on the training data and tested on the test data to assess their performance. We evaluated the models with the following metrics:

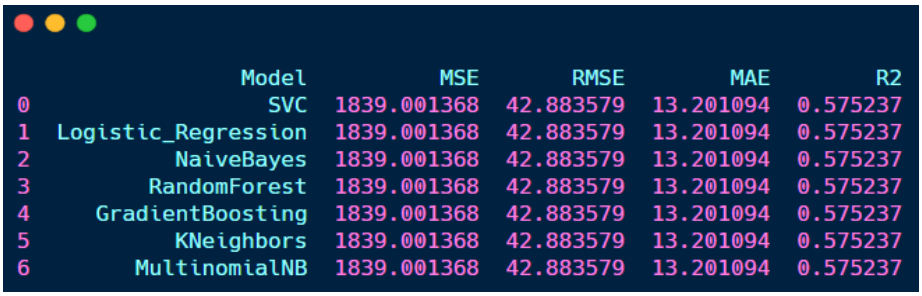
```
# Print accuracies
for model_name, accuracy in accuracies.items():
    print(f"{model_name} Accuracy: {accuracy}")

# Output :-
SVC Accuracy: 0.9028727770177839
Logistic_Regression Accuracy: 0.9211126310989513
NaiveBayes Accuracy: 0.9343365253077975
RandomForest Accuracy: 0.9179206566347469
GradientBoosting Accuracy: 0.8691290469676243
KNeighbors Accuracy: 0.8910168718650251
MultinomialNB Accuracy: 0.853625170998632
```

- The accuracy results show that the Naive Bayes model performed the best with an accuracy of 93.43%, followed by Logistic Regression at 92.11%. The Gradient Boosting model had the lowest accuracy at 86.91%, indicating that while most models performed well.

“ Calculate cross-validation, precision, recall, and F1-score for each class. These metrics are useful for understanding the performance of a classification model, especially when dealing with imbalanced classes. ”

Model Performance:

- All models have identical MSE, RMSE, MAE, and R2 values, indicating a possible issue with the evaluation process.



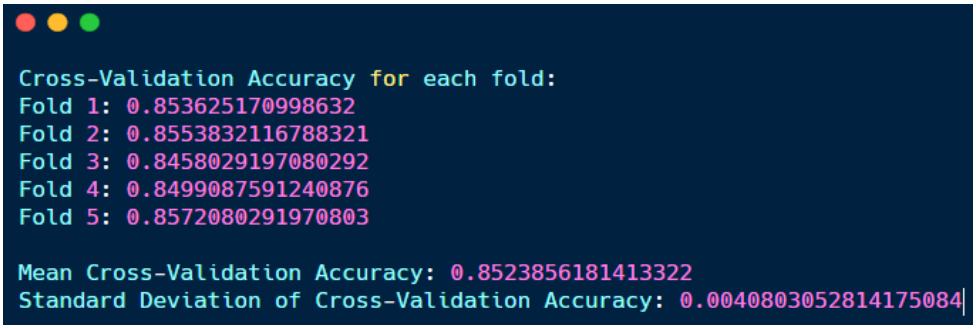
	Model	MSE	RMSE	MAE	R2
0	SVC	1839.001368	42.883579	13.201094	0.575237
1	Logistic_Regression	1839.001368	42.883579	13.201094	0.575237
2	NaiveBayes	1839.001368	42.883579	13.201094	0.575237
3	RandomForest	1839.001368	42.883579	13.201094	0.575237
4	GradientBoosting	1839.001368	42.883579	13.201094	0.575237
5	KNeighbors	1839.001368	42.883579	13.201094	0.575237
6	MultinomialNB	1839.001368	42.883579	13.201094	0.575237

Precision, Recall, and F1-score:

- The precision, recall, and F1-score results indicate significant performance variability across classes, with some classes performing perfectly.

Cross-Validation:

- The cross-validation results show consistent performance across the folds, with a mean accuracy of 85.24% and a low standard deviation of 0.41%, **indicating stable model performance.**



```
Cross-Validation Accuracy for each fold:
Fold 1: 0.853625170998632
Fold 2: 0.8553832116788321
Fold 3: 0.8458029197080292
Fold 4: 0.8499087591240876
Fold 5: 0.8572080291970803

Mean Cross-Validation Accuracy: 0.8523856181413322
Standard Deviation of Cross-Validation Accuracy: 0.0040803052814175084
```

Model Selection:

- Based on the best results among all models, the final model was chosen to be the Naive Bayes Classifier. In the future, it should be noted that the Naive Bayes model trained can be saved and used for making predictions.

Model Storage:

- The trained Naive Bayes model and the Label Encoder were saved using pickle to facilitate deployment and further predictions.
    - Naive Bayes Model File
    - Label Encoder File

******

4.4 Prediction System / Results

I have developed a recommendation system that diagnoses illnesses and offers tips according to symptoms given by users. More records were included to give details about symptoms, types of diseases that are there as they affect individuals differently, thus need different care, measures to avoid them, kind of drugs to administer, exercising techniques people should resort into, and the type diets recommended by specialists.

The model is used to predict the disease, as recommended by the tutor, after receiving the symptoms from the user. The prediction is followed by providing relevant recommendations including:

- Description of the predicted disease
- Precautions to be taken
- Recommended medications
- Suggested workouts
- Appropriate diets

Example Case:

```
# Get user input for symptoms
user_symptoms = ['fever', 'loss appetite', 'tiredness', 'headache', 'small', 'itchy blister']
# Predict the disease
predicted_disease = predict_disease(user_symptoms)
# Provide recommendations
provide_recommendations(predicted_disease)

# Output :-
=================predicted disease============
Chickenpox

=================description==================
Chickenpox is a highly contagious viral infection causing an itchy, blister-like rash.

=================precautions==================
1 :  Get vaccinated (Varicella vaccine)
2 :  Avoid contact with infected individuals
3 :  Keep nails short to prevent scratching
4 :  Use antihistamines and calamine lotion for itch relief

=================medications==================
1 :  Antiviral drugs
2 :  Pain relievers
3 :  Antihistamines
4 :  Calamine lotion
5 :  Antipyretics

=================workout======================
1 : Stay hydrated
2 : Rest and conserve energy
3 : Use calamine lotion for itching
4 : Monitor for symptoms
5 : Practice good hygiene

=================diets========================
1 : Comfort measures
2 : Hydration
3 : Antihistamines
4 : Calamine lotion
5 : Avoid scratching
```

Final Result :-

******

5. Phase-2 - Wellness Tracker

After completing the Phase 1 model, now I am moving towards Phase 2, in this phase I will generally build 3 models which will include pregnancy risk prediction, heart disease prediction and diabetic disease prediction. This will make AI Health Guard an advanced project. So let's move ahead…

5.1 Pregnancy Risk Prediction

This study aimed to develop a predictive model for an assessment of the level of risk during pregnancy based on associated health indicators. We used a dataset with information on age, Systolic BP, Diastolic BP, blood sugar level, body temperature, heart rate, among others. The study aimed at classification of the risk level as low, mid, or high using several machine learning algorithms in order to identify the most accurate model.

Data Overview

The dataset consists of 1,014 entries with 7 attributes: Age, SystolicBP, DiastolicBP, BS (Blood Sugar), BodyTemp (Body Temperature), HeartRate, and RiskLevel.

Key Descriptive Statistics:

- **Age**: Age of the patient
  - Range from 10 to 70 years, mean of 29.87 years.
- **SystolicBP**: Systolic blood pressure
  - Range from 70 to 160 mmHg, mean of 113.2 mmHg.
- **DiastolicBP**: Diastolic blood pressure
  - Range from 49 to 100 mmHg, mean of 76.46 mmHg.
- **BS**: Blood sugar levels
  - Range from 6.0 to 19.0 mmol/L, mean of 8.73 mmol/L.
- **BodyTemp**: Body temperature
  - Range from 98.0 to 103.0°F, mean of 98.67°F.
- **HeartRate**: Heart rate
  - Range from 7 to 90 bpm, mean of 74.3 bpm.
- **RiskLevel**: Risk level (low, mid, high)
  - Distribution - 406 low risk, 336 mid risk, 272 high risk.

Dataset Statistics

- **Shape:** 1014 records, 7 attributes
- **Risk Level Distribution:**
    - Low risk: 406
    - Mid risk: 336
    - High risk: 272

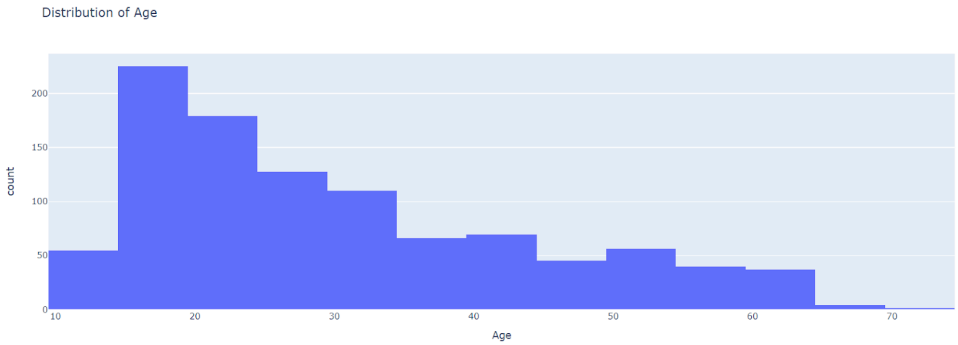
Data Preprocessing

- **Null Values:** The dataset contains no null values.
- **Outlier Removal:** An outlier with a heart rate of 7 was removed, reducing the dataset to 1012 records.
- **Multicollinearity:** Variance Inflation Factor (VIF) was calculated for systolic and diastolic blood pressure, resulting in the removal of systolic blood pressure due to high multicollinearity.

Data Visualization

From these, several visualizations were realized to better comprehend the distributions and relationships within the data:

- **Distribution of Age**:



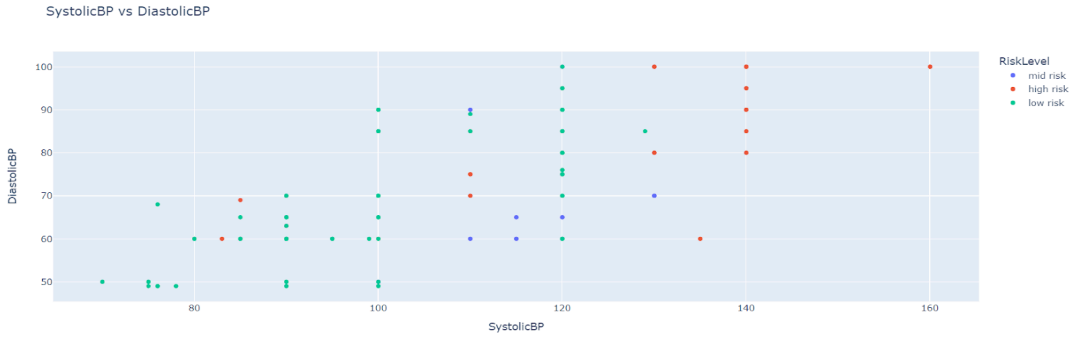
I analyzed that ,

- 10 - 15 Age group count 54
- 15 - 25 Age group count 404
- 25 - 35 Age group count 238
- 35 - 45 Age group count 135
- 45 - 55 Age group count 101

- 55 - 70 Age group count 81

So, 15 - 25 age group people get the maximum number of pregnancies.

- **SystolicBP vs DiastolicBP**:



  - Higher SystolicBP values and DiastolicBP values indicate an increased level of risk, with low risk mostly ranging from below 120/80 to high risk above 140/90.
  - Low risk is most common, followed by mid risk, with high risk being the least common but occurring at the highest blood pressure values.

Model Building

Data Splitting

- **Features:** Age, DiastolicBP, BS, BodyTemp, HeartRate
- **Target:** RiskLevel (encoded)
- **Train-Test Split:** 80% training, 20% testing

Model Performance

The accuracy of each model on the test set was as follows:

- **Accuracy Scores:**
  - Gradient Boosting Classifier: 77.83%
  - K-Nearest Neighbors: 68.97%
  - Logistic Regression: 62.07%
  - Decision Tree: 80.30%
  - Random Forest: 81.77%
  - SVM: 55.17%

Hyperparameter Tuning

GridSearchCV was used to fine-tune the RandomForestClassifier. The best parameters identified were:

```
Best parameters found:  {'criterion': 'entropy', 'max_depth': 20,
'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 200}
```

After tuning, the Random Forest model achieved an accuracy of 82.76%.

Model Comparison

Random Forest: Highest accuracy and robust performance after hyperparameter tuning.

Gradient Boosting Classifier: Second best with strong performance.

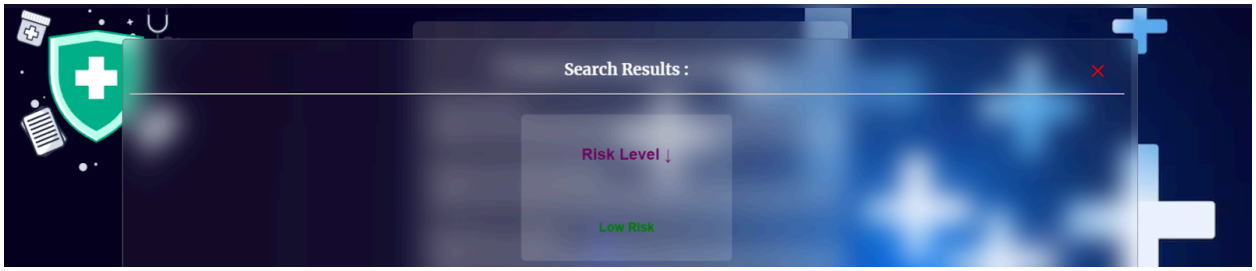
Decision Tree: Performed well but with potential overfitting issues.

Evaluation Metrics

- Precision, Recall, and F1 scores were also calculated for all models, with Random Forest showing the highest overall performance.

Results

The final Random Forest model was saved using Joblib for future use. A function was created to predict the risk level based on new data inputs.



******

5.2 Heart Disease Prediction

Heart disease is among the most serious global health problems, resulting in many deaths and numerous hospitalizations every year. Early prediction and diagnosis will significantly contribute to avoiding unfavorable outcomes as well as help improve the patient prognosis. This project aims to develop a predictive model for heart disease by using various machine learning algorithms and identifying the most effective one based on performance metrics.

Data Exploration and Preprocessing

- **Number of Entries:** 303
- **Number of Features:** 13
- **Target Variable:** Indicates whether a person has heart disease (1) or not (0).

Feature Information

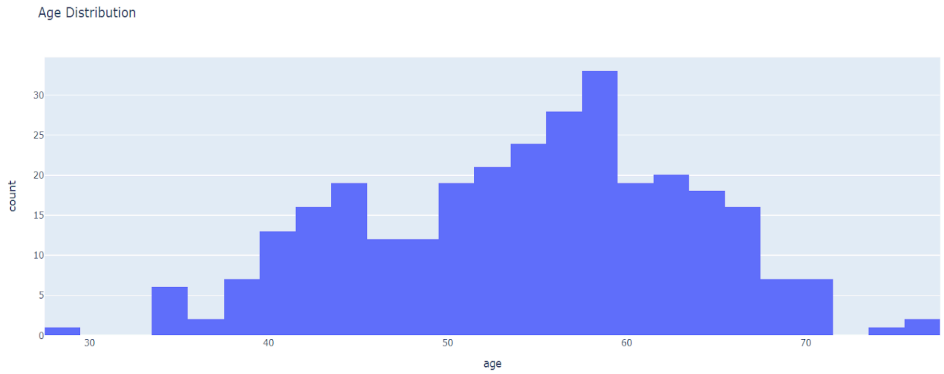
- **age:** Age of the patient
- **sex:** Sex of the patient (1 = male, 0 = female)
- **cp:** Chest pain type (0-3)
- **trestbps:** Resting blood pressure (in mmHg)
- **chol:** Serum cholesterol in mg/dl
- **fbs:** Fasting blood sugar > 120 mg/dl (1 = true, 0 = false)
- **restecg:** Resting electrocardiographic results (0-2)
- **thalach:** Maximum heart rate achieved
- **exang:** Exercise-induced angina (1 = yes, 0 = no)
- **oldpeak:** ST depression induced by exercise relative to rest
- **slope:** Slope of the peak exercise ST segment (0-2)
- **ca:** Number of major vessels colored by fluoroscopy (0-4)
- **thal:** Thalassemia (1 = normal, 2 = fixed defect, 3 = reversible defect)
- **target:** Presence of heart disease (1 = yes, 0 = no)

- The dataset is balanced with 165 entries indicating the presence of heart disease and 138 entries indicating the absence of heart disease.
- No missing values were found in the dataset.

Data Visualization

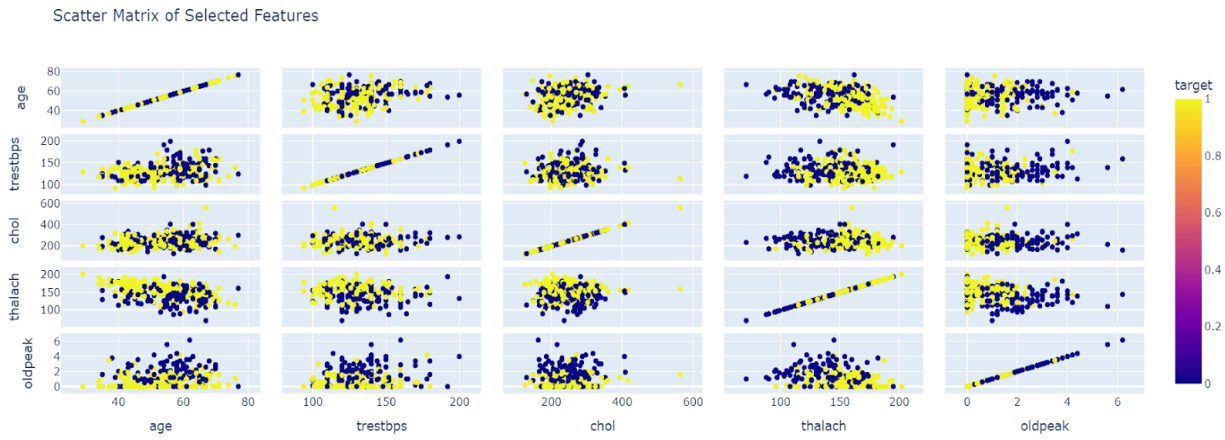
From these, several visualizations were realized to better comprehend the distributions and relationships within the data:

- **Age Distribution:**



  - 0 - 30 Age group count is 1
  - 30 - 40 Age group count is 15
  - 40 - 50 Age group count is 72
  - 50 - 60 Age group count is 125
  - 60 - 70 Age group count is 80
  - 70 - 80 Age group count is 9
  - So, the 50 - 60 age group get the maximum number of heart attacks.

- **Selected features to visualize pairwise relationships.**



  - Higher target values (yellow) are scattered across all features but are more concentrated in lower oldpeak values and higher thalach values.

- There is no strong linearity among the other features except for age that is uniformly distributed, forming a diagonal line.

Model Building

Data Splitting

The dataset was split into training and testing sets:

- **Training Set:** 80% of the data (242 entries)
- **Testing Set:** 20% of the data (61 entries)

Model Performance

The accuracy of each model on the test set was as follows:

- **Accuracy Scores:**
  - Gradient Boosting Classifier: 78.69%
  - K-Nearest Neighbors (KNN) Classifier**:** 59.02%
  - Logistic Regression: 85.25%
  - Decision Tree Classifier: 81.97%
  - Random Forest Classifier: 81.97%
  - Support Vector Machine (SVM): 57.38%

Hyperparameter Tuning

Hyperparameter tuning was performed on the Logistic Regression model using GridSearchCV. The best parameters identified were:

```
Best parameters found:  {'C': 1, 'penalty': 'l2', 'solver': 'liblinear'}
Best LogisticRegression Model after Hyperparameter Tuning:
Accuracy: 0.8524590163934426
MSE: 0.14754098360655737
MAE: 0.14754098360655737
```

After tuning, the Logistic Regression model achieved an accuracy of 85.25%.

Model Comparison

Logistic Regression: Highest accuracy and robust performance after hyperparameter tuning.

Random Forest: Second best with strong performance.

Decision Tree: Performed well but with potential overfitting issues.

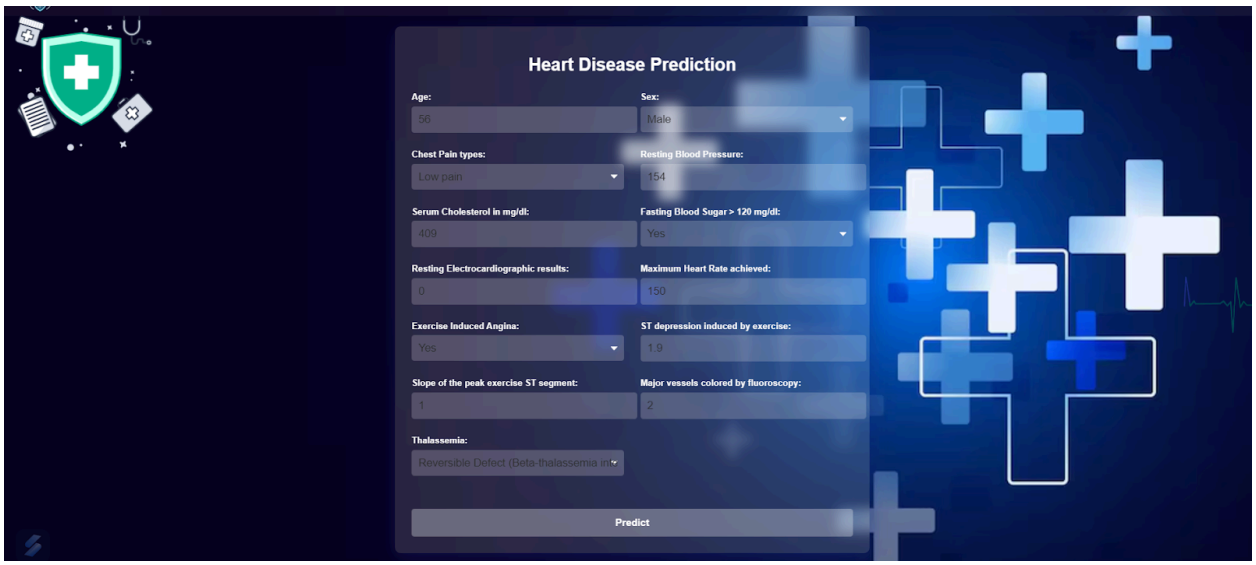
Cross-Validation

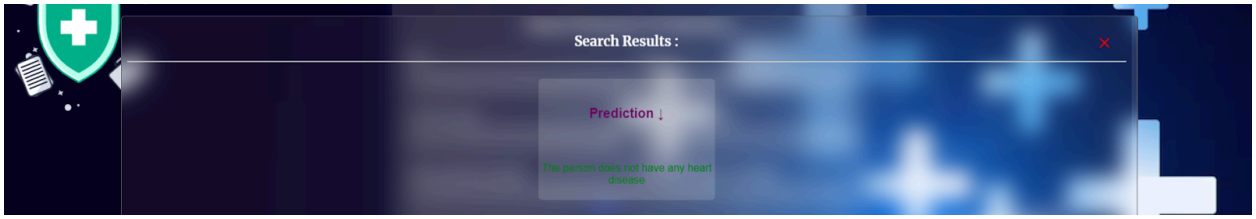
Cross-validation was performed on the best model (Logistic Regression) to ensure its robustness. The cross-validation scores were as follows:

- **Cross-Validation Scores:** [0.8689, 0.8033, 0.8033, 0.7833, 0.8667]
- **Average Cross-Validation Score:** 82.51%

Results

The final Logistic Regression model was saved using pickle for future use. A function was created to predict the risk level based on new data inputs.





******

5.3 Diabetic Disease Prediction

Diabetes being a chronic medical condition is experienced by millions of people globally. The prediction of diabetes very accurately could lead to effective patient management and care. This project will be focused on devising and evaluating different machine learning models for the possibility of a patient's risk of having diabetes based on their medical analysis.

Data Exploration and Preprocessing

- **Number of Entries:** 768
- **Number of Features:** 9
- The target variable 'Outcome' is binary, indicating whether a patient is diabetic (1) or non-diabetic (0).
- 500 samples are non-diabetic (65.1%)
- 268 samples are diabetic (34.9%)

Feature Information

The features represent various medical attributes such as:

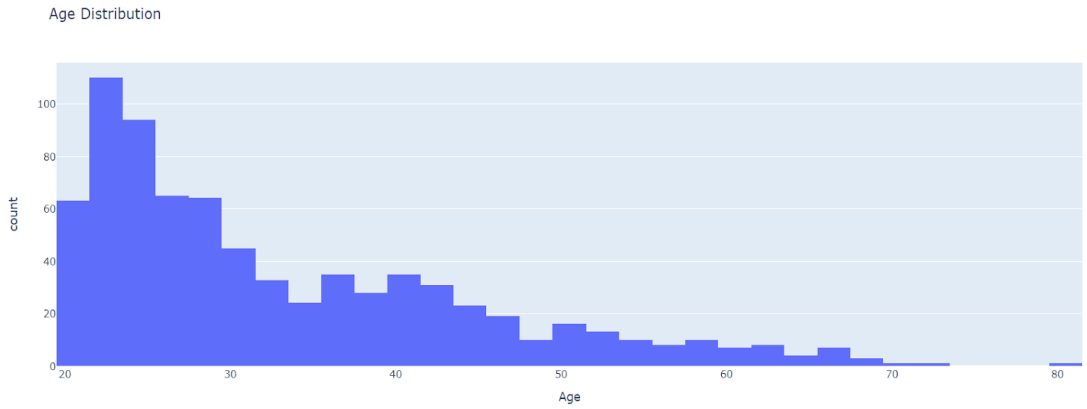
- **Pregnancies**
- **Glucose**
- **BloodPressure**
- **SkinThickness**
- **Insulin**
- **BMI**
- **DiabetesPedigreeFunction**
- **Age**
- **Outcome (0: Non-Diabetic, 1: Diabetic)**

- The dataset contains no missing values.
- Distribution of the Outcome variable shows 500 non-diabetic and 268 diabetic cases.

Data Visualization

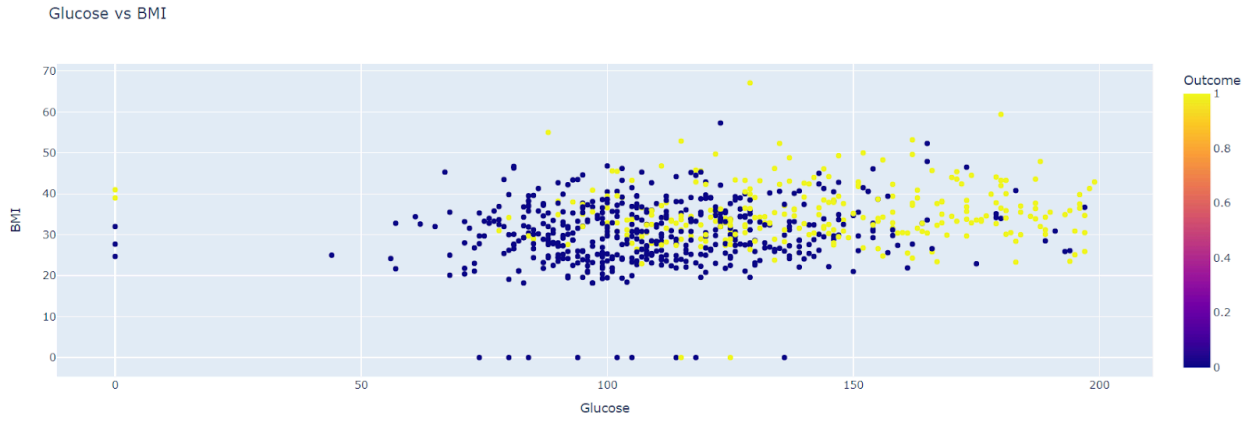
From these, several visualizations were realized to better comprehend the distributions and relationships within the data:

- **Age Distribution:**



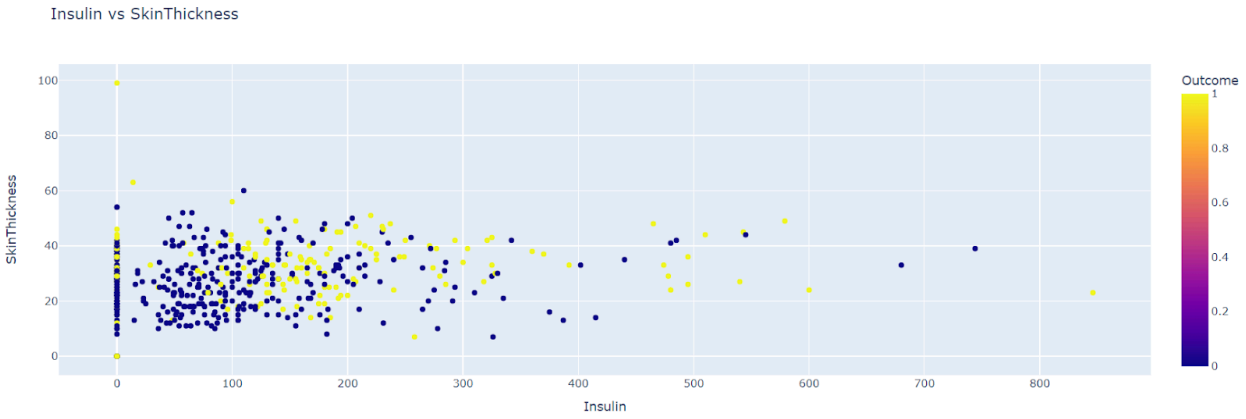
  - 20 - 30 Age Group count is 396
  - 30 - 40 Age Group count is 165
  - 40 - 50 Age Group count is 118
  - 50 - 60 Age Group count is 57
  - 60 - 70 Age Group count is 29
  - 70 - 80 Age Group count is 2
  - So, the 20 - 30 age group gets the maximum number of diabetic patients.

- **Glucose vs BMI**



  - Higher glucose levels show a greater spread of outcomes (more yellow), indicating a higher likelihood of positive outcomes as glucose increases.
  - BMI has a wide range of values across all glucose levels, with no strong linear correlation evident between BMI and outcome.

- **Insulin vs SkinThickness**



  - A potential correlation between insulin levels and skin thickness is suggested, where data points with lower insulin levels appear to be more densely packed in the lower left quadrant of the graph.
  - The results are indicated using a color gradient, as different results are displayed depending on the intensity of the color.

Model Building

Data Splitting

- The dataset was split into training and test sets with an 80-20 ratio.
- Data standardization was performed using StandardScaler.

Model Performance

The accuracy of each model on the test set was as follows:

- **Accuracy Scores:**
  - Logistic Regression: 75.3%
  - Decision Tree: 74.7%
  - Random Forest: 74.0%
  - Support Vector Machine: 73.4%
  - K-Nearest Neighbors: 69.5%

Hyperparameter Tuning

Hyperparameter tuning was conducted for Logistic Regression using GridSearchCV, resulting in the following best parameters:

```
Best parameters found:  {'C': 0.1, 'penalty': 'l1', 'solver': 'liblinear'}
Best LogisticRegression Model after Hyperparameter Tuning:
Accuracy: 0.7597402597402597
MSE: 0.24025974025974026
```

After tuning, the Logistic Regression model achieved an accuracy of 75.97% .

Model Comparison

Models were evaluated based on train and test scores, precision, recall, and F1 score:

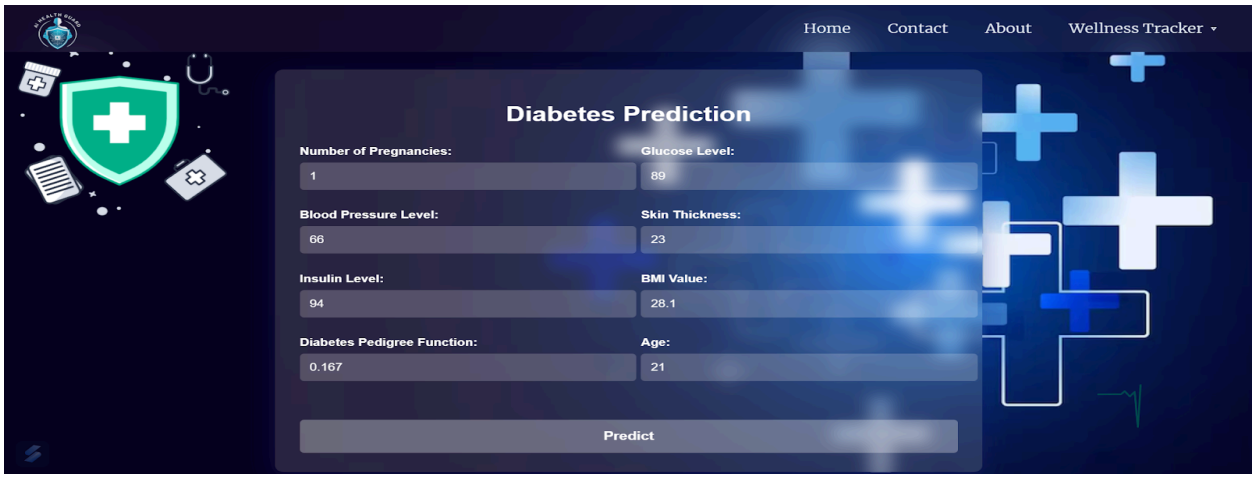
- Logistic Regression and Random Forest achieved the highest test score of 75.97%.
- K-Nearest Neighbors had the lowest test score of 69.5%.

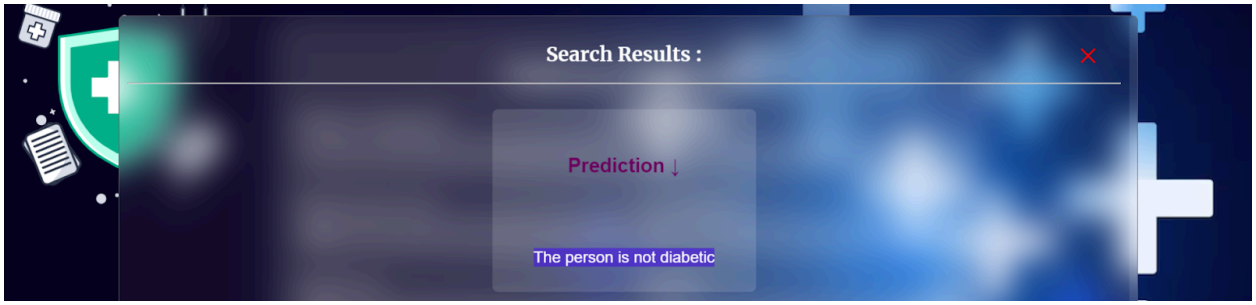
Evaluation Metrics

- Precision, Recall, and F1 scores were also calculated for all models, with Random Forest showing the highest overall performance.

Results

The final Logistic Regression model was saved using pickle for future use. A function was created to predict the risk level based on new data inputs.





******

6. Development and Backend Integration

This section details which I write about the backend processes used to build this system. The backend implementation involves integrating various datasets, normalizing data, and utilizing a pre-trained machine learning model for accurate disease prediction.

Phase - 1 Backend Explain

Data Integration and Management:-

- It also integrates datasets for results that can be consistent and accurate. They include symptoms, precautions, workouts, description, medication, and diets.
- The datasets are loaded from CSV files. The columns and the data in the dataset are normalized to have uniformity.
- Column names are standardized, and all diseases' names are changed to lowercase and extra whitespace removed.

Model Utilization :-

It incorporates a Support Vector Classifier model for the prediction of diseases from symptoms. The following major components help in integration with the model:

- **Model Loading:** It loads a pre-trained model in pickle format named (`svc.pkl`).
- **Label encoding:** The (`label_encoder.pkl`) is the component used to map back predicted labels to the original disease names.

Prediction Workflow:-

- Users input their symptoms through a web interface. These symptoms are processed to match the model's feature set.
- The symptoms are transformed into a binary format, where each symptom corresponds to a feature in the model.
- The SVC model predicts the disease based on the input symptoms. The label encoder then converts the predicted disease label to its original form.

Some Code Snippets :-

```python
# Function to predict disease based on symptoms
def predict_disease(symptoms):
    symptoms_dict = {symptom: 0 for symptom in svc.feature_names_in_}
    for symptom in symptoms:
        symptom = symptom.strip().lower()
        if symptom in symptoms_dict:
            symptoms_dict[symptom] = 1

    input_data = pd.DataFrame([symptoms_dict])
    predicted_disease = svc.predict(input_data)
    disease_name = le.inverse_transform(predicted_disease)[0].strip().lower()
```

etc...

Phase - 2 Backend Explain

This section describes the development and backend functionality of a health prediction system designed to assess risks related to pregnancy, heart disease, and diabetes based on user inputs.

System Overview:-

The health prediction system comprises three primary components:

1. Pregnancy Risk Prediction
2. Heart Disease Prediction
3. Diabetes Prediction

Each component uses a specific machine learning model to predict health risks based on user-provided data.

Model Loading:-

Models are loaded during the application initialization phase:

- **Pregnancy Model**: Loaded from the file `` `pregnancy_model.pkl` ``.
- **Heart Disease Model**: Loaded from the file `` `Heart.sav` ``.
- **Diabetes Disease Model**: Loaded from the file `` `Diabetes.sav` ``.

Prediction Functionality:-

The system provides a user interface for inputting health-related parameters and returns risk assessments based on the models.

1. **Pregnancy Risk Prediction :-**
    - **Input Parameters**: Age, diastolic blood pressure, blood sugar, body temperature, and heart rate.
    - **Output**: Risk level categorized as Low, Medium, or High.
2. **Heart Disease Prediction :-**
    - **Input Parameters**: Age, sex, chest pain type, resting blood pressure, cholesterol, fasting blood sugar, rest ECG, maximum heart rate, exercise-induced angina, ST depression, slope of ST segment, number of major vessels, and thalassemia.
    - **Output**: Prediction of whether the person has heart disease.
3. **Diabetes Disease Prediction:-**
    - **Input Parameters**: Pregnancies, glucose, blood pressure, skin thickness, insulin, BMI, diabetes pedigree function, and age.
    - **Output**: Prediction of whether the person is diabetic.

User Interaction:-
The system has user-friendly web forms to facilitate easy entry of health parameters by the user, with results of the prediction shown immediately. Each module of prediction can be a stand-alone module but will have a common interface only for easy use by the users.

Some Code Snippets :-

```
def diabetes():
    prediction_text = None
    if request.method == 'POST':
        data = [request.form['Pregnancies'],
            request.form['Glucose'],
            request.form['BloodPressure'],
            request.form['SkinThickness'],
            request.form['Insulin'],
            request.form['BMI'],
            request.form['DiabetesPedigreeFunction'],
            request.form['Age']]
        input_data = [data]
        with warnings.catch_warnings():
            warnings.simplefilter("ignore")
            prediction = diabetic_model.predict(input_data)
        if prediction[0] == 1:
            prediction_text = 'The person is diabetic'
        else:
            prediction_text = 'The person is not diabetic'

    return render_template('diabetes.html', prediction_text=prediction_text)
```

etc...

Health Dashboard Development :-

This section details the creation of a health dashboard using Flask, designed for visualizing and analyzing disease and symptom data.

Dataset Utilization :- Two datasets are used:

- **Original Dataset**: For generating initial visualizations.
- **Encoded Dataset (df_v1)**: For advanced visualizations.

Dashboard Interface :-
The Flask application provides a user-friendly web interface:

- **Disease Frequency Bar Chart**: Accessible via the main dashboard route.
- **Symptom Word Cloud**: Embedded within the dashboard for immediate symptom analysis.
- **Top Symptoms Frequency Plot**: Displayed for quick reference to common symptoms.
- **Pie Chart and Bubble Chart**: Offer additional perspectives on symptom data.

- **Symptom Distribution per Disease**: Allows selection of a disease to view corresponding symptom distributions.
- **Disease-Symptom Network Graph**: Provides an interactive visualization of disease and symptom connections.
- **3D Scatter Plot and Sankey Diagram**: Enable detailed exploration of symptom relationships and flows.

Visualization Techniques :-

Visualization tools and techniques employed include:

- **Plotly**: For interactive charts and plots.
- **WordCloud**: For generating word clouds.
- **NetworkX**: For creating network graphs.
- **Matplotlib and Seaborn**: For additional visualizations and data handling.

User Interaction :-

- **Disease Selection**: Dropdown menu for analyzing specific diseases.
- **Dynamic Updates**: Charts and graphs update based on user selections.
- **Tooltips and Hover Information**: Enhance user experience by providing detailed information on hover.

Some Code Snippets :-

```
# Parallel Categories Diagram
  fig6 = px.parallel_categories(df,
                color_continuous_scale=px.colors.sequential.Inferno, title="")
  graph6 = fig6.to_html(full_html=False)

# Flows Between Diseases and Symptoms
  nodes = list(df_v1['Disease'].unique()) + list(df_v1.columns[1:10])
  links = []
  for disease in df_v1['Disease'].unique():
      for symptom in df_v1.columns[1:10]:
          count = df_v1[(df_v1['Disease'] == disease) &
                        (df_v1[symptom] ==1)].shape[0]
          if count > 0:
              links.append({'source': nodes.index(symptom),
                            'target': nodes.index(disease), 'value': count})

  fig8 = go.Figure(go.Sankey(node=dict(pad=15, thickness=20,
                             line=dict(color="black", width=0.5), label=nodes ),
                             link=dict(source=[link['source'] for link in links],
                             target=[link['target'] for link in links],
                             value=[link['value'] for link in links])))
  fig8.update_layout(title_text="", font_size=10)
  graph8 = fig8.to_html(full_html=False)
```

etc...

******

7. Further Enhancement and Conclusion

A number of the potential ways in which this could be carried forward are:

- **Scalability Improvements:** Further optimization and scalability improvements would be in place to work with bigger data sizes and more complicated kinds of scenarios. This may involve considering more advanced algorithms or distributed computing techniques.
- **Features Upgrades:** Incorporating new features that will increase the functions of the system and make it friendlier to work with. For instance, real-time data processes or more advanced data visualization tools can be introduced.
- **Automation and AI Integration:** The use of machine learning and artificial intelligence will automate multiple dimensions of the project, making it more accurate and efficient.

Conclusion

Conclusively, The AI Health Guard project represents a significant endeavor in utilizing data science and machine learning techniques to empower individuals in managing their health effectively. By leveraging advanced algorithms and personalized recommendations, the system aims to enhance healthcare outcomes and promote overall well-being.

Overall, AI Health Guard is set to be a groundbreaking addition to the field of health technology, offering a proactive and personalized approach to health maintenance and disease prevention.

Project Code :- https://github.com/alokchoudhary05/AI-HEALTH-GUARD

Live Project Link :- https://aihealthguard-io.onrender.com

(When you open the website by clicking on the link, please wait couple of minutes, it takes some seconds to load the website.)

******

8. References

[1] Kaggle, "Retrieved data from," Mar. 15, 2024 - May 5, 2024. [Online]. Available: Link .

[2] Python Software Foundation, "Python 3.12.4 — Welcome to Python.org." [Online]. Available: Link .

[3] Scikit-learn, "scikit-learn: machine learning in Python — scikit-learn 0.24.0 documentation." [Online]. Available: Link .

[4] Google Colaboratory, "Google Colaboratory," Google Research, 2024. [Online]. Available: Link .

[5] NumPy, "NumPy — (Version: 1.26)." [Online]. Available: Link .

[6] Pandas, "Pandas — Version: 2.2.2 — Python Data Analysis Library." [Online]. Available: Link .

[7] Matplotlib, "Matplotlib 3.9.0 — Matplotlib: Visualization with Python." [Online]. Available: Link .

[8] Plotly as px, "Plotly — (Dash v2.17.0)." [Online]. Available: Link .

[9] Krish Naik, "Machine Learning: Algorithms, Statistics: Stats For Data Science," YouTube, 2022. [Online]. Available: Link , Link .

[10] Geeks for Geeks, "Machine Learning Algorithms, Statistics in Maths," Geeks for Geeks, 2021. [Online]. Available: Link, Link .

[11] Flask Documentation, "Welcome to Flask — Flask Documentation (3.0.x)." [Online]. Available: Link .

[12] Streamlit, "Streamlit," Streamlit Inc., 2024. [Online]. Available: Link .

[13] "Process and Workflow Image," GitHub Image. [Online]. Available: Link .

[14] "AI vs ML vs DS vs DL image," created by me.

******