Chapter 1: Introduction as well as Objective

Early as well as precise disease identification is essential for successful treatment as well as better patient outcomes in today's healthcare environment. With just five input symptoms, our cutting-edge multi-disease prediction system uses machine learning to forecast a variety of common ailments like infections (fungal) as well as jaundice, along with other common diseases like fungal infection, typhoid, jaundice, as well as many more. With the help of strong machine learning algorithms as well as an easy-to-use Streamlit-based user interface, this system enables users—from consumers to medical professionals—to rapidly as well as accurately evaluate any health hazards. Our approach seeks to close the gap between sophisticated medical diagnostics as well as easily available health information by fusing cutting-edge predictive analytics with an intuitive design, opening the door for proactive healthcare management.

Objectives:

The primary objectives of this project are:

- Develop as well as train various machine learning algorithms as well as models for prediction of disease like Diabetes, Parkinson's, Cardiovascular, jaundice, typhoid, as well as many more.
- Implement a user-friendly web interface in order to input medical parameters as well as obtain the necessary predictions.
- Combine predictions for multiple diseases into a cohesive platform.
- Demonstrating the value of machine learning in the facilitation of early diagnosis as well as proactive health management.
- Enhancing the model's robustness through advanced techniques such as outlier removal, data standardization, as well as ensemble learning.

Chapter 2: System Analysis

Problem Definition:

Cardiovascular Disorder continues to be one of the leading causes of mortality globally. While manual diagnosis based on lab data as well as medical history can be subjective, delayed, or unreliable, early prediction can save lives. A machine learning system uses clinical data to predict if the patient is prone to a Cardiovascular Disorder, Parkinson's disease, diabetes, typhoid, infections (fungal), or jaundice. The algorithm looks for patterns suggestive of these disorders by evaluating features like blood pressure, glucose, cholesterol, BMI, bilirubin levels, speech data, movement sensor readings, skin lesion photos, as well as epidemiological data. The cardiovascular risks of Cardiovascular Disorder, the neurological symptoms of Parkinson's disease, the metabolic imbalances of diabetes, the infectious indicators of typhoid, the systemic or dermatological signs of infections (fungal), as well as the liver-related problems of jaundice can all be detected early with this method. The approach improves patient outcomes by more precision in diagnosis as well as expediting intervention across different diseases by processing huge datasets as well as reducing human error.

Inputs:

- **Cardiovascular Disorder Prediction:**
    - Age
    - Sex
    - Chest Pain Type
    - RestingBP
    - Cholesterol
    - Fasting Blood Sugar
    - Resting ECG
    - Max Heart Rate
    - Exercise Angina

- Oldpeak
- ST Slope

- **Parkinson's Disease Prediction:**
  - MDVP:Fo(Hz)
  - MDVP:Fhi(Hz)
  - MDVP:Flo(Hz)
  - MDVP:Jitter(%)
  - MDVP:Jitter(Abs)
  - MDVP:RAP
  - MDVP:PPQ
  - Jitter:DDP
  - MDVP:Shimmer
  - MDVP:Shimmer(dB)
  - Shimmer:APQ3
  - Shimmer:APQ5
  - MDVP:APQ
  - Shimmer:DDA
  - NHR
  - HNR
  - RPDE
  - DFA
  - spread1
  - spread2
  - D2
  - PPE

- **Diabetes Prediction:**
  - Number of Pregnancies

    - Glucose Level
    - Blood Pressure Value
    - Skin Thickness Value
    - Insulin Level
    - BMI Value
    - Diabetes Pedigree Function Value
    - Age of the Person

- **Disease Prediction Based on Symptoms:** The person can choose 5 main symptoms from a variety of symptoms. These symptoms include: 'back pain', 'constipation', 'abdominal pain', 'diarrhoea', 'mild fever', 'yellow urine', 'yellowing of eyes', 'acute liver failure', 'fluid overload', 'swelling of stomach', 'swelled lymph nodes', 'malaise', 'blurred as well as distorted vision', 'phlegm', 'throat irritation', 'redness of eyes', 'sinus pressure', 'runny nose', 'congestion', 'chest pain', 'weakness in limbs', 'fast heart rate', 'pain during bowel movements', 'pain in anal region', 'bloody stool', 'irritation in anus', 'neck pain', 'dizziness', 'cramps', 'bruising', 'obesity', 'swollen legs', 'swollen blood vessels', 'puffy face as well as eyes', 'enlarged thyroid', 'brittle nails', 'swollen extremeties', 'excessive hunger', 'extra marital contacts', 'drying as well as tingling lips', 'slurred speech', 'knee pain', 'hip joint pain', 'muscle weakness', 'stiff neck', 'swelling joints', 'movement stiffness', 'spinning movements', 'loss of balance', 'unsteadiness', 'weakness of one body side', 'loss of smell', 'bladder discomfort', 'continuous feel of urine', 'passage of gases', 'internal itching', 'toxic look (typhos)', 'depression', 'irritability', 'muscle pain', 'altered sensorium', 'red spots over body', 'belly pain', 'abnormal menstruation', 'watering from eyes', 'increased appetite', 'polyuria', 'family history', 'mucoid sputum', 'rusty sputum', 'lack of concentration', 'visual disturbances', 'receiving blood transfusion', 'receiving unsterile injections', 'coma', 'stomach bleeding', 'distention of abdomen', 'history of alcohol consumption', 'blood in sputum', 'prominent veins on calf', 'palpitations', 'painful walking', 'pus filled pimples', 'blackheads', 'skin peeling', 'silver like dusting', 'small dents in nails', 'inflammatory nails', 'blister', 'red sore around nose', 'yellow crust ooze'.
    - Symptom 1
    - Symptom 2

- Symptom 3
- Symptom 4
- Symptom 5

Output:

- Cardiovascular Disorder: Based on the inputs, the model will give the output whether the person has any chances of developing a Cardiovascular Disorder or not.
- Parkinson's Disease: Based on the inputs, the model will give the output whether the person has any chances of developing Parkinson's disease or not.
- Diabetes: Based on the inputs provided, the model will give the result whether the person has chances of developing diabetes or not.
- Disease Prediction Based on Symptoms: Based on the inputs provided by the user, the model will give 3 predictions of the possible diseases. The diseases that the model can predict are: 'Fungal infection', 'Allergy', 'GERD', 'Chronic cholestasis', 'Drug Reaction', 'Peptic ulcer disease', 'AIDS', 'Diabetes ', 'Gastroenteritis', 'Bronchial Asthma', 'Hypertension ', 'Migraine', 'Cervical spondylosis', 'Paralysis (brain hemorrhage)', 'Jaundice', 'Malaria', 'Chicken pox', 'Dengue', 'Typhoid', 'hepatitis A', 'Hepatitis B', 'Hepatitis C', 'Hepatitis D', 'Hepatitis E', 'Alcoholic hepatitis', 'Tuberculosis', 'Common Cold ','Pneumonia', 'Dimorphic hemorrhoids(piles)', 'Heart attack', 'Varicose veins', 'Hypothyroidism', 'Hyperthyroidism', 'Hypoglycemia', 'Osteoarthristis', 'Arthritis', '(vertigo) Paroymsal Positional Vertigo', 'Acne', 'Urinary tract infection', 'Psoriasis', 'Impetigo'

Potential Users:

- Data Scientists / Developers (during development)
- Medical professionals (during deployment, via a UI or app)
- Patients (indirect users via apps)

Chapter 3: Data Flow Diagram (DFD)

The application system's needs as well as the system's structural flow are the main topics of the data flow diagram. It is a visual depiction of the flow of data through a process or system. Using symbols like rectangles, circles, as well as arrows, it shows how data is entered, processed, saved, as well as generated; it makes no mention of the sequence or time of these operations.

In software engineering, systems analysis, as well as business process management, DFDs are widely used to understas well as as well as improve systems.

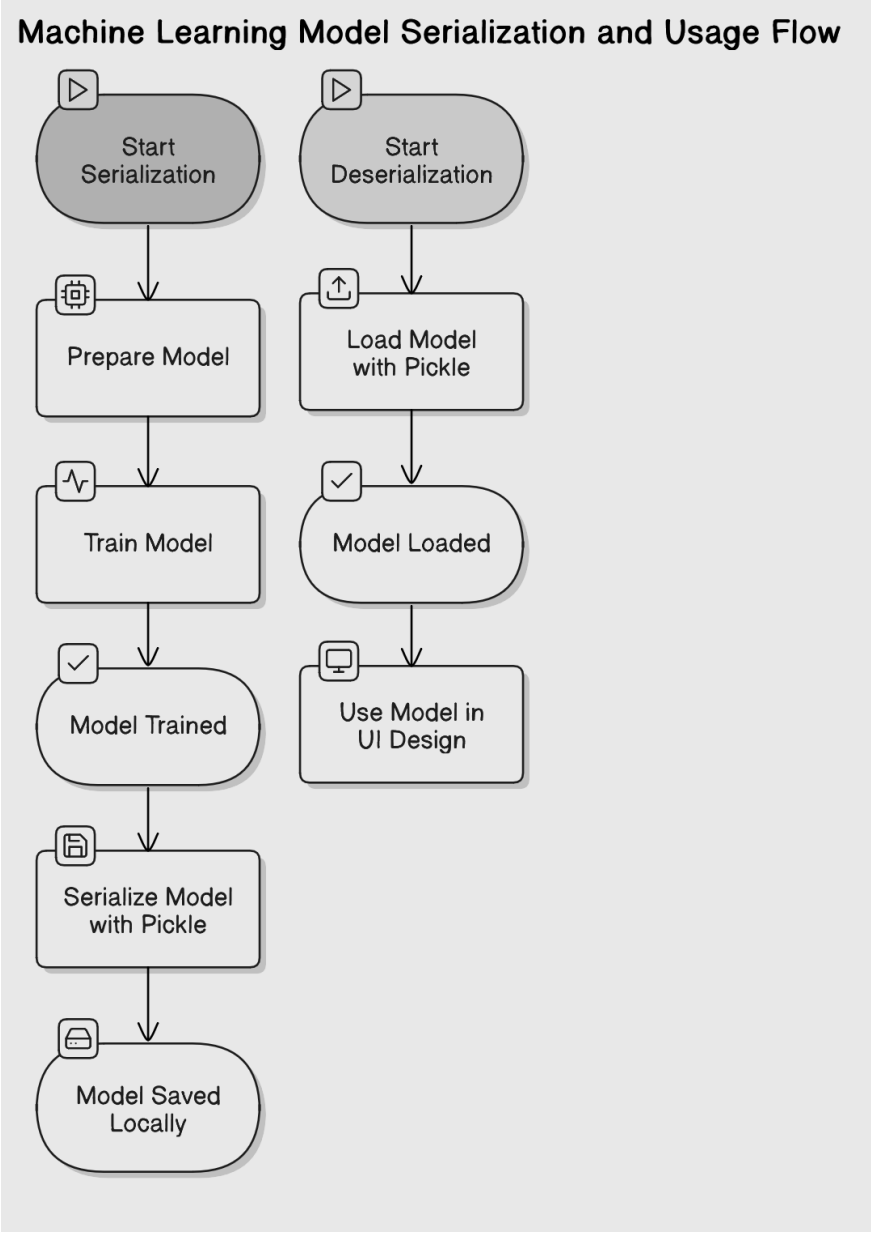


Fig. Machine Learning Model Serialization as well as Usage Flow

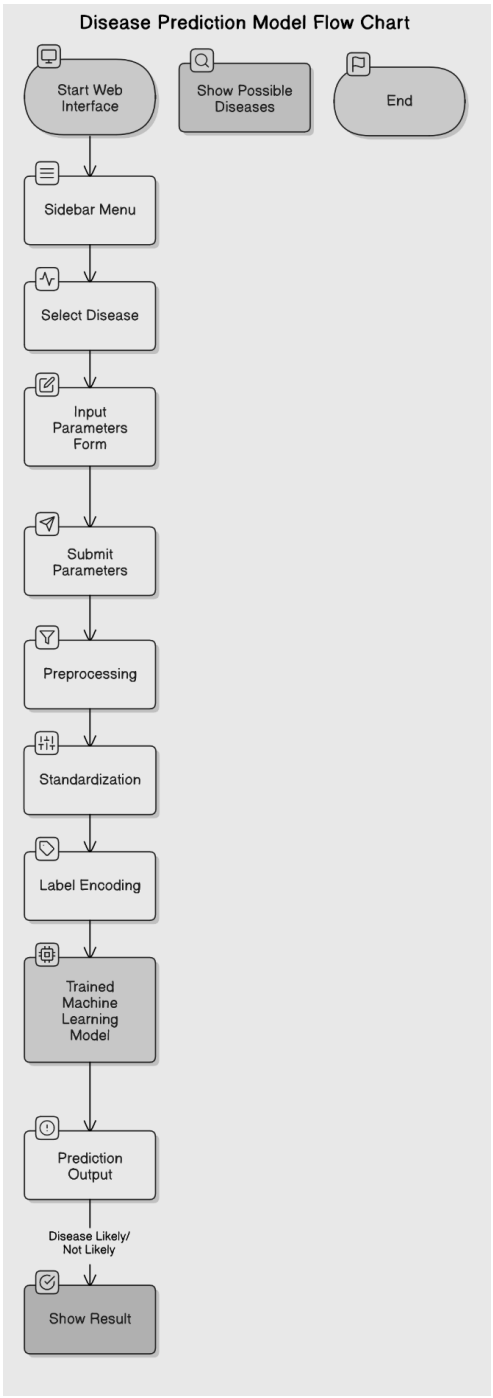


Fig. Disease Prediction Model Flow Chart

Chapter 4: Requirement Specification

Working:

The Multiple Disease Prediction System predicts a number of medical conditions using four machine learning models. The predictors for diabetes as well as Parkinson's disease classify individuals based on numerical health metrics using a Support Vector Classifier (SVC) with a linear kernel. A complex stacking classifier for Cardiovascular Disorder is created by combining the predictions of three base learners: Random Forest, XGBoost, as well as Logistic Regression. The output is then routed to a final logistic regression estimator. The benefits of many models are used in this ensemble approach to improve forecast accuracy. The symptoms-based disease prediction model employs three distinct algorithms: Decision Tree, Random Forest, as well as Naive Bayes. A dataset that connected symptoms to possible illnesses was used to train each algorithm.

Once these models have been trained as well as saved locally, Python's pickle module serializes the learned model objects so they may be used again without needing to be retrained. Streamlit, an open-source Python framework that facilitates the rapid creation of interactive web applications directly from Python scripts, is used to design the user interface (UI) of this system. Even non-programmers may use Streamlit with ease due to its extremely user-friendly design. Users interact with the system by entering their personal health information in dropdowns, sliders, or input forms, which may include their age, blood pressure, glucose level, or specific symptoms, depending on the prediction module they are using. Each of the various components that comprise the user interface corresponds to a different illness prediction model.

By clicking the "Predict" button, users can load a model into memory as well as enter data into the Streamlit interface. The system preprocesses the input data, including scaling as well as encoding, before feeding it into the trained model, as well as the results are clearly labeled on the user interface, allowing users to quickly understas well as their potential health state. Three predictions from the Decision Tree, Random Forest, as well as Naive Bayes models are provided in the section on symptoms-based prediction, as well as one prediction is returned for the diabetes, Parkinson's, as well as Cardiovascular Disorder modules.

Programming Language: Python

Libraries Used:

- Pas well asas
- Numpy
- Pickle
- sklearn.model_selection.train_test_split
- sklearn.preprocessing.StandardScaler
- sklearn.preprocessing.LabelEncoder
- sklearn.ensemble.RandomForestClassifier
- xgboost.XGBClassifier
- sklearn.linear_model.LogisticRegression
- sklearn.ensemble.StackingClassifier
- sklearn.metrics.classification_report
- sklearn.metrics.roc_curve
- sklearn.metrics.auc
- matplotlib.pyplot
- seaborn
- sklearn.svm
- sklearn.metrics.accuracy_score
- sklearn.svm.SVC
- sklearn.tree.DecisionTreeClassifier
- sklearn.naive_bayes.GaussianNB
- sklearn.model_selection.cross_val_score
- streamlit
- streamlit_option_menu.

Machine Learning Models Used:

- **Random Forest:** In order to increase efficiency (accuracy) as well as decrease overfitting, this ensemble machine learning approach constructs many decision trees as

well as then aggregates their predictions, either by voting or averaging. This technique works well for jobs involving regression as well as classification.

- **XGBoost:** It is a gradient boosting approach that successively improves decision trees to reduce mistakes with great accuracy as well as efficiency. This method works well for applications involving ranking, regression, as well as classification.
- **Logistic Regression:** It is a statistical technique that uses a logical function to forecast probability for binary or multiclass classification. For data that is linearly separable, this approach works well.
- **Stacking Classifier:** It is an ensemble method that combines predictions from many base classes, such as Random Forest or SVM, with the aid of a meta-classifier to enhance generalization as well as overall performance.
- **Support Vector Machine (SVM):** It's an algorithm for classification. SVM maximizes the margin between classes by determining the best hyperplane to divide them. Using kernel methods, it works well with high-dimensional as well as non-linear data.
- **Decision Tree:** The model is based on trees. To make judgments, a decision tree divides data into branches according to feature criteria. Although it may be used for both regression as well as classification, if it is not pruned, it is prone to overfitting.
- **Gaussian Naive Bayes:** Based on the Bayes theorem, it is a probability classifier that assumes features are conditionally independent as well as have a Gaussian distribution. Simple as well as quick categorization jobs are appropriate for it.

Development Environment: Model development was done in a Jupyter notebook. Deployment was done using Python scripts. UI design was done using the Spyder IDE.

Datasets Used:

- **Diabetes Dataset:** Pima Indians Diabetes Dataset (available at Kaggle)
    - Total Features: 9
    - Labels: 0 (Non-diabetic), 1 (Diabetic)
    - Size: 768 records
- **Parkinson's Disease Dataset:** Voice Measurements Dataset (available at Kaggle)

    - Total Features: 22
    - Labels: 0 (Healthy), 1 (Parkinson's)
    - Size: 195 records
- **Cardiovascular Disorder Dataset:** Clevelas well as Cardiovascular Disorder Dataset (available at Kaggle)
    - Total Features: 12
    - Labels: 0 (Healthy), 1 (Cardiovascular Disorder)
    - Size: 918
- **Symptoms Based Disease Prediction:** (available at Kaggle)
    - Total Features: 133
    - Labels: 40 different prognosis
    - Size: 4961

Chapter 5: System Architecture as well as Methodology

This system follows a modular approach:

User Input → Data Preprocessing → Prediction Model → Output Result

Each illness prediction model makes use of a web interface made using Python's Streamlit package to take particular medical parameters. It then uses preprocessing techniques like label encoding and standardization before processing inputs using a machine learning model that has been developed. The result tells the user what the disease may be based on the symptoms or forecasts if the user is likely to have a certain ailment.

The online interface, which was constructed with streamlit, has input fields for entering parameters and a sidebar menu for choosing the target ailment.

Methodology:

Date Preprocessing: One fundamental and important stage in transforming raw data into some meaningful information is data preparation. The majority of the time, data may be noisy, repetitious, or incomplete. These problems may be fixed by preprocessing the data, which can then be applied to the creation of machine learning models.

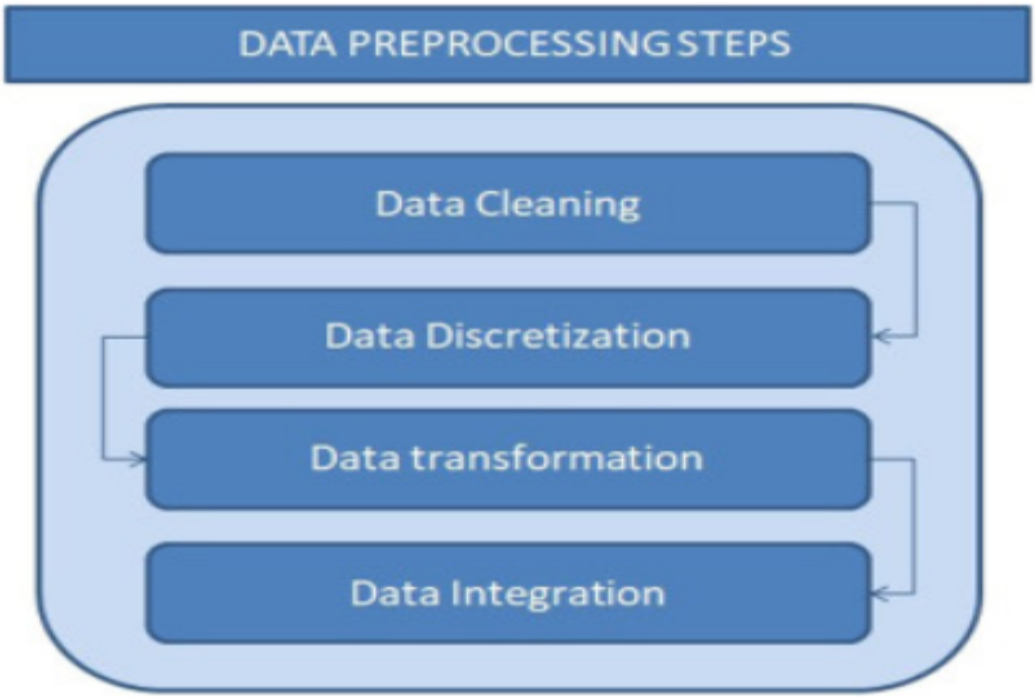


Fig. Data Preprocessing Steps

There are four major steps in data preprocessing. These steps are:

- Data Cleaning: Reliable machine learning and analysis results depend on clean data, which calls for repairing formatting problems, deleting duplicates, and fixing flaws and inconsistencies in the data, such as missing numbers.
- Data Discretization: By converting continuous data into discrete intervals, data discretization enhances model performance and facilitates understanding. Binning, histogram analysis, and clustering-based techniques are examples of common techniques.
- Data Transformation: Data transformation improves model accuracy and convergence by transforming data into formats that are appropriate for analysis or modeling, such as normalization, standardization, encoding, and mathematical functions.
- Data Integration: Particularly in remote databases and cloud services, data integration combines data from several sources for analysis, resolving schema mismatches and inconsistencies using ETL procedures and data warehousing.

The following steps were applied in the ‘Multiple Disease Prediction System’ for data preprocessing:

- Missing values: Descriptive statistics were utilized to either eliminate or modify any missing values in the dataset.
- Outlier removal: Using the box plot and interquartile range, outliers in the dataset were identified and addressed.
- Encoding: LabelEncoder was used to encode the datasets' categorical variables.
- Standardization: To guarantee uniform scaling across features, StandardScaler was used to standardize numerical features.
- Data Splitting: Datasets were split into training (80%) as well as testing (20%) sets.
- SMOTE (Synthetic Minority Oversampling Technique): SMOTE was used in order to deal with any class imbalance in the datasets.

Model Training:

- Diabetes Prediction: Applied Support Vector Classifier (SVC) with a linear kernel.
- Parkinson’s Disease Prediction: Applied Support Vector Classifier (SVC) with a linear kernel.
- Cardiovascular Disorder Prediction: Applied a Stacking Classifier combining Random Forest, XGBoost, as well as Logistic Regression, with Logistic Regression as the final estimator.
- Symptoms Based Disease Prediction: Applied three distinct disease prediction models: Decision Tree, Random Forest, as well as Naive Bayes, each trained independently as well as evaluated on test data, resulting in a total of three predictions.

Model Accuracy:

Disease	**Model(s) Used**	**Accuracy**
Cardiovascular Disorder Prediction	Stacking Classifier combining Random	84.9 % (cross-validation accuracy)

	Forest, XGBoost, as well as Logistic Regression Final Estimator: Logistic Regression	
Diabetes Prediction	SVC	78 %
Parkinson's Disease Prediction	SVC	87.2 %
Symptoms Based Disease Prediction	Decision Tree, Random Forest, as well as Naive Bayes	92.68 % (for all three models)

UI Design: A Multiple Disease Prediction System User Interface was developed using the open-source Python framework Streamlit. Data scientists and AI/ML engineers may construct dynamic data apps with less code thanks to this easy-to-use platform. Pickle, a package that offers binary protocols for serializing and de-serializing Python object structures, was used to save the models locally. Following that, these models were loaded for UI design.

Chapter 6: Code Implementation with Screenshots

Diabetes Prediction Model:

```
import numpy as np
import pas well asas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score

diabetes_dataset = pd.read_csv('diabetes.csv')

X = diabetes_dataset.drop(columns = 'Outcome', axis=1)
Y = diabetes_dataset['Outcome']

scaler = StandardScaler()

scaler.fit(X)

sandardized_data = scaler.transform(X)

X = sandardized_data
Y = diabetes_dataset['Outcome']

X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size = 0.2, stratify=Y, Random_state=2)

classifier = svm.SVC(kernel='linear')
```

```
classifier.fit(X_train, Y_train)

X_train_prediction = classifier.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)

X_test_prediction = classifier.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)

input_data = (5,166,72,19,175,25.8,0.587,51)
input_data_as_numpy_array = np.asarray(input_data)
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)
std_data = scaler.transform(input_data_reshaped)
print(std_data)
prediction = classifier.predict(std_data)
print(prediction)
if (prediction[0] == 0):
  print('The person is not diabetic')
else:
  print('The person is diabetic')

import pickle
filename = "diabetes_model.sav"
pickle.dump(classifier, open(filename, "wb"))

filename = "diabetes_scaler.sav"
pickle.dump(scaler, open(filename, "wb"))
```

Parkinson's Prediction Model:

```
import pas well asas as pd
import numpy as np

data = pd.read_csv("dataset.csv")

X=data.drop(columns = ['name', 'status'], axis=1)
y=data['status']

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2, Random_state=2)

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

scaler.fit_transform(X_train)
scaler.fit_transform(X_test)

from sklearn.svm import SVC

model = SVC(kernel='linear')

model.fit(X_train, y_train)

pred_train = model.predict(X_train)
```

```
from sklearn.metrics import accuracy_score

train_acc = accuracy_score(y_train, pred_train)

pred_test = model.predict(X_test)

input_data =
(122.4,148.65,113.819,0.00968,0.00008,0.00465,0.00696,0.01394,0.06134,0.626,0.03134,0.045
18,0.04368,0.09403,0.01929,19.085,0.458359,0.819521,-4.075192,0.33559,2.486855,0.368674)
input_array = np.asarray(input_data)
reshaped_data = input_array.reshape(1,-1)
std_data = scaler.transform(reshaped_data)
prediction = model.predict(std_data)
if prediction[0]==0:
        print("Healthy")
else:
        print("You might have Parkinson Disease")

import pickle
filename = "parkinsons_model.sav"
pickle.dump(model, open(filename, "wb"))
filename = "parkinsons_scaler.sav"
pickle.dump(scaler, open(filename, "wb"))
```

Cardiovascular Disorder Prediction:

```
import pas well asas as pd
import numpy as np
from sklearn.model_selection import train_test_split, cross_val_score
```

```
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import StackingClassifier
from sklearn.metrics import classification_report, roc_curve, auc
import matplotlib.pyplot as plt
import seaborn as sns

heart_failure = pd.read_csv('heart.csv')

def validate_data(df):
    print("Data Validation Report:")

    print("\nAge range:", df['Age'].min(), "to", df['Age'].max())

    print("\nUnique Sex values:", df['Sex'].unique())

    print("\nUnique ChestPainType values:", df['ChestPainType'].unique())

    print("\nRestingBP range:", df['RestingBP'].min(), "to", df['RestingBP'].max())

    print("\nCholesterol range:", df['Cholesterol'].min(), "to", df['Cholesterol'].max())

    print("\nUnique FastingBS values:", df['FastingBS'].unique())

    print("\nUnique RestingECG values:", df['RestingECG'].unique())
```

```
    print("\nMaxHR range:", df['MaxHR'].min(), "to", df['MaxHR'].max())

    print("\nUnique ExerciseAngina values:", df['ExerciseAngina'].unique())

    print("\nUnique ST_Slope values:", df['ST_Slope'].unique())

validate_data(heart_failure)

def remove_outliers(df, columns):
    df_clean = df.copy()

    for column in columns:
    Q1 = df_clean[column].quantile(0.25)
    Q3 = df_clean[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    outliers = ((df_clean[column] < lower_bound) | (df_clean[column] > upper_bound))
    print(f"\nNumber of outliers in {column}: {outliers.sum()}")

    df_clean = df_clean[~outliers]

    return df_clean

numerical_columns = ['Age', 'RestingBP', 'Cholesterol', 'MaxHR', 'Oldpeak']

plt.figure(figsize=(15, 5))
```

```
heart_failure[numerical_columns].boxplot()
plt.title('Boxplots Before Outlier Removal')
plt.xticks(rotation=45)
plt.show()

heart_failure_clean = remove_outliers(heart_failure, numerical_columns)

plt.figure(figsize=(15, 5))
heart_failure_clean[numerical_columns].boxplot()
plt.title('Boxplots After Outlier Removal')
plt.xticks(rotation=45)
plt.show()

print(f"\nOriginal dataset size: {len(heart_failure)}")
print(f"Dataset size after outlier removal: {len(heart_failure_clean)}")

def clean_cholesterol_improved(df):
    df_clean = df.copy()

    df_clean = df_clean[df_clean['Cholesterol'] >= 100]

    Q1 = df_clean['Cholesterol'].quantile(0.25)
    Q3 = df_clean['Cholesterol'].quantile(0.75)
    IQR = Q3 - Q1

    lower_bound = max(Q1 - 1.5 * IQR, 130)
    upper_bound = min(Q3 + 1.5 * IQR, 400)
```

```
        cholesterol_outliers = ((df_clean['Cholesterol'] < lower_bound) |
                (df_clean['Cholesterol'] > upper_bound))

        print(f"Number of Cholesterol outliers: {cholesterol_outliers.sum()}")
        print(f"Range for Cholesterol: {lower_bound:.2f} to {upper_bound:.2f}")

        df_clean = df_clean[~cholesterol_outliers]
        return df_clean

heart_failure_clean = clean_cholesterol_improved(heart_failure)

plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
bp1 = plt.boxplot(heart_failure['Cholesterol'],
                patch_artist=True,
                medianprops=dict(color="orange"),
                boxprops=dict(facecolor='lightblue'))
plt.title('Cholesterol Before Cleaning')
plt.ylabel('Cholesterol (mg/dl)')

plt.subplot(1, 2, 2)
bp2 = plt.boxplot(heart_failure_clean['Cholesterol'],
                patch_artist=True,
                medianprops=dict(color="orange"),
                boxprops=dict(facecolor='lightblue'))
plt.title('Cholesterol After Cleaning')
plt.ylabel('Cholesterol (mg/dl)')
```

```
plt.tight_layout()
plt.show()

print("\nCholesterol Statistics After Cleaning:")
print(heart_failure_clean['Cholesterol'].describe())
print(f"\nOriginal dataset size: {len(heart_failure)}")
print(f"Dataset size after cleaning: {len(heart_failure_clean)}")

df = heart_failure_clean.copy()

le = LabelEncoder()

categorical_columns = ['Sex', 'ChestPainType', 'RestingECG', 'ExerciseAngina', 'ST_Slope']
for column in categorical_columns:
        df[column] = le.fit_transform(df[column])

X = df.drop('HeartDisease', axis=1)
y = df['HeartDisease']

numerical_columns = ['Age', 'RestingBP', 'Cholesterol', 'MaxHR', 'Oldpeak']
scaler = StandardScaler()
X[numerical_columns] = scaler.fit_transform(X[numerical_columns])

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, Random_state=42)

base_models = [
        ('rf', RandomForestClassifier(Random_state=42)),
```

```
        ('xgb', XGBClassifier(Random_state=42)),
        ('lr', LogisticRegression(Random_state=42))
]

stacking = StackingClassifier(
        estimators=base_models,
        final_estimator=LogisticRegression(),
        cv=5,
        stack_method='predict_proba'
)

stacking.fit(X_train, y_train)

def get_detailed_risk_assessment(patient_data, probability):
        risk_prob = probability * 100

        if risk_prob < 20:
        risk_level = "Low"
        recommendation = "Maintain healthy lifestyle as well as regular check-ups"
        timeframe = "Annual check-up recommended"
        elif risk_prob < 40:
        risk_level = "Moderate-Low"
        recommendation = "Continue regular monitoring with lifestyle modifications"
        timeframe = "Follow-up in 6 months"
        elif risk_prob < 60:
        risk_level = "Moderate-High"
        recommendation = "Schedule consultation with healthcare provider"
        timeframe = "Follow-up within 1 month"
```

```
elif risk_prob < 80:
risk_level = "High"
recommendation = "Prompt medical evaluation needed"
timeframe = "Follow-up within 1 week"
else:
risk_level = "Very High"
recommendation = "Urgent medical attention required"
timeframe = "Immediate medical attention"

risk_factors = []
concerns = []

if patient_data['Age'] > 60:
risk_factors.append("Advanced age")
elif patient_data['Age'] > 45:
concerns.append("Age-related risk")

if patient_data['RestingBP'] >= 140:
risk_factors.append("High blood pressure")
elif patient_data['RestingBP'] >= 120:
concerns.append("Elevated blood pressure")

if patient_data['Cholesterol'] >= 240:
risk_factors.append("High cholesterol")
elif patient_data['Cholesterol'] >= 200:
concerns.append("Borderline cholesterol")

if patient_data['ST_Slope'] == 'Down':
```

```
        risk_factors.append("Abnormal ST slope")
        elif patient_data['ST_Slope'] == 'Flat':
        concerns.append("Flat ST slope")

        if patient_data['ExerciseAngina'] == 'Y':
        risk_factors.append("Exercise-induced angina")

        if patient_data['ChestPainType'] == 'ASY':
        risk_factors.append("Asymptomatic chest pain")

        return risk_level, recommendation, timeframe, risk_factors, concerns

def predict_comprehensive(patient_data, model, scaler):
        patient_df = pd.DataFrame([patient_data])

        categorical_columns = ['Sex', 'ChestPainType', 'RestingECG', 'ExerciseAngina',
'ST_Slope']
        le = LabelEncoder()
        for column in categorical_columns:
        patient_df[column] = le.fit_transform(patient_df[column])

        numerical_columns = ['Age', 'RestingBP', 'Cholesterol', 'MaxHR', 'Oldpeak']
        patient_df[numerical_columns] = scaler.transform(patient_df[numerical_columns])

        prediction = model.predict(patient_df)
        probability = model.predict_proba(patient_df)[0][1]

        risk_level, recommendation, timeframe, risk_factors, concerns =
get_detailed_risk_assessment(patient_data, probability)
```

```
        print("\n=== Comprehensive Cardiovascular Disorder Risk Assessment ===")
        print("\nPatient Profile:")
        print(f"Age: {patient_data['Age']}, Sex: {patient_data['Sex']}")
        print(f"Blood Pressure: {patient_data['RestingBP']} mm Hg")
        print(f"Cholesterol: {patient_data['Cholesterol']} mm/dl")
        print(f"Max Heart Rate: {patient_data['MaxHR']}")

        print("\nCritical Indicators:")
        print(f"Chest Pain Type: {patient_data['ChestPainType']}")
        print(f"Exercise Angina: {'Present' if patient_data['ExerciseAngina']=='Y' else
'Absent'}")
        print(f"ST Depression: {patient_data['Oldpeak']}")
        print(f"ST Slope: {patient_data['ST_Slope']}")

        print("\nRisk Assessment:")
        print(f"Risk Level: {risk_level}")
        print(f"Cardiovascular Disorder Probability: {probability:.1%}")

        if risk_factors:
        print("\nMajor Risk Factors:")
        for factor in risk_factors:
        print(f"• {factor}")

        if concerns:
        print("\nAreas of Concern:")
        for concern in concerns:
        print(f"• {concern}")
```

```
        print(f"\nRecommendation: {recommendation}")
        print(f"Timeframe: {timeframe}")

        if risk_level in ["High", "Very High"]:
        print("\nWarning: Multiple high-risk factors detected. Immediate medical consultation advised.")

y_pred_final = stacking.predict(X_test)
print("\nModel Performance Metrics:")
print(classification_report(y_test, y_pred_final))

cv_scores = cross_val_score(stacking, X, y, cv=5)
print(f"\nCross-validation accuracy: {cv_scores.mean():.3f} (+/- {cv_scores.std() * 2:.3f})")

import pickle

with open('stacking_model.pkl', 'wb') as f:
        pickle.dump(stacking, f)

with open('scaler.pkl', 'wb') as f:
        pickle.dump(scaler, f)

label_encoders = {}
categorical_columns = ['Sex', 'ChestPainType', 'RestingECG', 'ExerciseAngina', 'ST_Slope']

for column in categorical_columns:
        le = LabelEncoder()
```

```
        le.fit(heart_failure_clean[column])
        label_encoders[column] = le

with open('label_encoders.pkl', 'wb') as f:
        pickle.dump(label_encoders, f)

print("Model, scaler, as well as encoders have been saved successfully!")
```

Symptoms Based Disease Prediction:

```
import pas well asas as pd
import numpy as np
import pickle

l1 = ['back_pain', 'constipation', 'abdominal_pain', 'diarrhoea', 'mild_fever', 'yellow_urine',
'yellowing_of_eyes', 'acute_liver_failure', 'fluid_overload', 'swelling_of_stomach',
'swelled_lymph_nodes', 'malaise', 'blurred_as well as_distorted_vision', 'phlegm',
'throat_irritation',
'redness_of_eyes', 'sinus_pressure', 'runny_nose', 'congestion', 'chest_pain', 'weakness_in_limbs',
'fast_heart_rate', 'pain_during_bowel_movements', 'pain_in_anal_region', 'bloody_stool',
'irritation_in_anus', 'neck_pain', 'dizziness', 'cramps', 'bruising', 'obesity', 'swollen_legs',
'swollen_blood_vessels', 'puffy_face_as well as_eyes', 'enlarged_thyroid', 'brittle_nails',
'swollen_extremeties', 'excessive_hunger', 'extra_marital_contacts', 'drying_as well
as_tingling_lips',
'slurred_speech', 'knee_pain', 'hip_joint_pain', 'muscle_weakness', 'stiff_neck', 'swelling_joints',
'movement_stiffness', 'spinning_movements', 'loss_of_balance', 'unsteadiness',
'weakness_of_one_body_side', 'loss_of_smell', 'bladder_discomfort', 'continuous_feel_of_urine',
'passage_of_gases', 'internal_itching', 'toxic_look_(typhos)', 'depression', 'irritability',
'muscle_pain', 'altered_sensorium', 'red_spots_over_body', 'belly_pain', 'abnormal_menstruation',
```

'watering_from_eyes', 'increased_appetite', 'polyuria', 'family_history', 'mucoid_sputum', 'rusty_sputum', 'lack_of_concentration', 'visual_disturbances', 'receiving_blood_transfusion', 'receiving_unsterile_injections', 'coma', 'stomach_bleeding', 'distention_of_abdomen', 'history_of_alcohol_consumption', 'blood_in_sputum', 'prominent_veins_on_calf', 'palpitations', 'painful_walking', 'pus_filled_pimples', 'blackheads', 'skin_peeling', 'silver_like_dusting', 'small_dents_in_nails', 'inflammatory_nails', 'blister', 'red_sore_around_nose', 'yellow_crust_ooze']

```
df = pd.read_csv("Training.csv")

df.replace({'prognosis': {
        'Fungal infection':0,'Allergy':1,'GERD':2,'Chronic cholestasis':3,'Drug Reaction':4,
        'Peptic ulcer diseae':5,'AIDS':6,'Diabetes ':7,'Gastroenteritis':8,'Bronchial Asthma':9,
        'Hypertension ':10,'Migraine':11,'Cervical spondylosis':12,'Paralysis (brain hemorrhage)':13,
        'Jaundice':14,'Malaria':15,'Chicken pox':16,'Dengue':17,'Typhoid':18,'hepatitis A':19,
        'Hepatitis B':20,'Hepatitis C':21,'Hepatitis D':22,'Hepatitis E':23,'Alcoholic hepatitis':24,
        'Tuberculosis':25,'Common Cold':26,'Pneumonia':27,'Dimorphic hemmorhoids(piles)':28,
        'Heart attack':29,'Varicose veins':30,'Hypothyroidism':31,'Hyperthyroidism':32,'Hypoglycemia':33,
        'Osteoarthristis':34,'Arthritis':35,'(vertigo) Paroymsal  Positional Vertigo':36,'Acne':37,
        'Urinary tract infection':38,'Psoriasis':39,'Impetigo':40
}}, inplace=True)

X = df[l1]
y = np.ravel(df[['prognosis']])

tr = pd.read_csv("Testing.csv")
```

```
tr.replace({'prognosis': {
        'Fungal infection':0,'Allergy':1,'GERD':2,'Chronic cholestasis':3,'Drug Reaction':4,
        'Peptic ulcer diseae':5,'AIDS':6,'Diabetes ':7,'Gastroenteritis':8,'Bronchial Asthma':9,
        'Hypertension ':10,'Migraine':11,'Cervical spondylosis':12,'Paralysis (brain hemorrhage)':13,
        'Jaundice':14,'Malaria':15,'Chicken pox':16,'Dengue':17,'Typhoid':18,'hepatitis A':19,
        'Hepatitis B':20,'Hepatitis C':21,'Hepatitis D':22,'Hepatitis E':23,'Alcoholic hepatitis':24,
        'Tuberculosis':25,'Common Cold':26,'Pneumonia':27,'Dimorphic hemmorhoids(piles)':28,
        'Heart attack':29,'Varicose veins':30,'Hypothyroidism':31,'Hyperthyroidism':32,'Hypoglycemia':33,
        'Osteoarthristis':34,'Arthritis':35,'(vertigo) Paroymsal  Positional Vertigo':36,'Acne':37,
        'Urinary tract infection':38,'Psoriasis':39,'Impetigo':40
}}, inplace=True)

X_test = tr[l1]
y_test = np.ravel(tr[['prognosis']])

from sklearn.tree import DecisionTreeClassifier
clf_dt = DecisionTreeClassifier()
clf_dt.fit(X, y)

from sklearn.metrics import accuracy_score
y_pred_dt = clf_dt.predict(X_test)
print("Decision Tree Accuracy:", accuracy_score(y_test, y_pred_dt))

with open('decision_tree_model.pkl', 'wb') as f:
        pickle.dump(clf_dt, f)

from sklearn.ensemble import RandomForestClassifier
```

```
clf_rf = RandomForestClassifier()
clf_rf.fit(X, y)

y_pred_rf = clf_rf.predict(X_test)
print("Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf))

with open('Random_forest_model.pkl', 'wb') as f:
        pickle.dump(clf_rf, f)

from sklearn.naive_bayes import GaussianNB
clf_nb = GaussianNB()
clf_nb.fit(X, y)

y_pred_nb = clf_nb.predict(X_test)
print("Naive Bayes Accuracy:", accuracy_score(y_test, y_pred_nb))

with open('naive_bayes_model.pkl', 'wb') as f:
        pickle.dump(clf_nb, f)

print("All models trained as well as saved successfully!")
```

Streamlit Code for UI Design:

```
import pickle
import streamlit as st
from streamlit_option_menu import option_menu
import numpy as np
import pas well asas as pd
```

```
diabetes_model = pickle.load(open("C:/Users/shubh/OneDrive/Desktop/Multiple Disease Prediction/saved models/diabetes_model.sav", 'rb'))
diabetes_scaler = pickle.load(open("C:/Users/shubh/OneDrive/Desktop/Multiple Disease Prediction/saved models/diabetes_scaler.sav", 'rb'))

parkinsons_model = pickle.load(open("C:/Users/shubh/OneDrive/Desktop/Multiple Disease Prediction/saved models/parkinsons_model.sav", 'rb'))
parkinsons_scaler = pickle.load(open("C:/Users/shubh/OneDrive/Desktop/Multiple Disease Prediction/saved models/parkinsons_scaler.sav", 'rb'))

decision_tree_model = pickle.load(open("C:/Users/shubh/OneDrive/Desktop/Multiple Disease Prediction/saved models/decision_tree_model.pkl", 'rb'))
Random_forest_model = pickle.load(open("C:/Users/shubh/OneDrive/Desktop/Multiple Disease Prediction/saved models/Random_forest_model.pkl", 'rb'))
naive_bayes_model = pickle.load(open("C:/Users/shubh/OneDrive/Desktop/Multiple Disease Prediction/saved models/naive_bayes_model.pkl", 'rb'))

stacking_model = pickle.load(open('C:/Users/shubh/OneDrive/Desktop/Multiple Disease Prediction/saved models/stacking_model.pkl', 'rb'))
scaler = pickle.load(open('C:/Users/shubh/OneDrive/Desktop/Multiple Disease Prediction/saved models/scaler.pkl', 'rb'))
label_encoders = pickle.load(open('C:/Users/shubh/OneDrive/Desktop/Multiple Disease Prediction/saved models/label_encoders.pkl', 'rb'))

autism_model = pickle.load(open('C:/Users/shubh/OneDrive/Desktop/Multiple Disease Prediction/saved models/autism_model.pkl', 'rb'))
autism_encoders = pickle.load(open('C:/Users/shubh/OneDrive/Desktop/Multiple Disease Prediction/saved models/autism_encoders.pkl', 'rb'))
```

```
l1 = ['back_pain', 'constipation', 'abdominal_pain', 'diarrhoea', 'mild_fever', 'yellow_urine',
        'yellowing_of_eyes', 'acute_liver_failure', 'fluid_overload', 'swelling_of_stomach',
'swelled_lymph_nodes',
        'malaise', 'blurred_as well as_distorted_vision', 'phlegm', 'throat_irritation',
'redness_of_eyes', 'sinus_pressure',
        'runny_nose', 'congestion', 'chest_pain', 'weakness_in_limbs', 'fast_heart_rate',
'pain_during_bowel_movements',
        'pain_in_anal_region', 'bloody_stool', 'irritation_in_anus', 'neck_pain', 'dizziness',
'cramps', 'bruising',
        'obesity', 'swollen_legs', 'swollen_blood_vessels', 'puffy_face_as well as_eyes',
'enlarged_thyroid', 'brittle_nails',
        'swollen_extremeties', 'excessive_hunger', 'extra_marital_contacts', 'drying_as well
as_tingling_lips', 'slurred_speech',
        'knee_pain', 'hip_joint_pain', 'muscle_weakness', 'stiff_neck', 'swelling_joints',
'movement_stiffness',
        'spinning_movements', 'loss_of_balance', 'unsteadiness', 'weakness_of_one_body_side',
'loss_of_smell',
        'bladder_discomfort', 'continuous_feel_of_urine', 'passage_of_gases', 'internal_itching',
'toxic_look_(typhos)',
        'depression', 'irritability', 'muscle_pain', 'altered_sensorium', 'red_spots_over_body',
'belly_pain',
        'abnormal_menstruation', 'watering_from_eyes', 'increased_appetite', 'polyuria',
'family_history', 'mucoid_sputum',
        'rusty_sputum', 'lack_of_concentration', 'visual_disturbances',
'receiving_blood_transfusion', 'receiving_unsterile_injections',
        'coma', 'stomach_bleeding', 'distention_of_abdomen', 'history_of_alcohol_consumption',
'blood_in_sputum',
        'prominent_veins_on_calf', 'palpitations', 'painful_walking', 'pus_filled_pimples',
'blackheads', 'skin_peeling',
```

```
        'silver_like_dusting', 'small_dents_in_nails', 'inflammatory_nails', 'blister',
'red_sore_around_nose', 'yellow_crust_ooze']

disease_map = {
        0: 'Fungal infection', 1: 'Allergy', 2: 'GERD', 3: 'Chronic cholestasis', 4: 'Drug Reaction',
        5: 'Peptic ulcer diseae', 6: 'AIDS', 7: 'Diabetes', 8: 'Gastroenteritis', 9: 'Bronchial Asthma',
        10: 'Hypertension', 11: 'Migraine', 12: 'Cervical spondylosis', 13: 'Paralysis (brain
hemorrhage)',
        14: 'Jaundice', 15: 'Malaria', 16: 'Chicken pox', 17: 'Dengue', 18: 'Typhoid', 19: 'hepatitis
A',
        20: 'Hepatitis B', 21: 'Hepatitis C', 22: 'Hepatitis D', 23: 'Hepatitis E', 24: 'Alcoholic
hepatitis',
        25: 'Tuberculosis', 26: 'Common Cold', 27: 'Pneumonia', 28: 'Dimorphic
hemmorhoids(piles)', 29: 'Heart attack',
        30: 'Varicose veins', 31: 'Hypothyroidism', 32: 'Hyperthyroidism', 33: 'Hypoglycemia', 34:
'Osteoarthristis',
        35: 'Arthritis', 36: '(vertigo) Paroymsal  Positional Vertigo', 37: 'Acne', 38: 'Urinary tract
infection',
        39: 'Psoriasis', 40: 'Impetigo'
}



with st.sidebar:
        selected = option_menu("Multiple Disease Prediction System",

                ["Diabetes Prediction",
                        "Parkisons Prediction",
                        "Disease Prediction using ML Models",
```

```
                    "Cardiovascular Disorder Prediction"],

                icons = ['activity', 'person', 'hospital', 'heart-pulse-fill'],

                    default_index = 0)

if (selected == 'Diabetes Prediction'):
    st.title("Diabetes Prediction using ML")


    col1, col2, col3 = st.columns(3)

    with col1:
    Pregnancies = st.text_input("Number of Pregnancies")

    with col2:
    Glucose = st.text_input("Glucose Level")

    with col3:
    BloodPressure = st.text_input("Blood Pressure Value")

    with col1:
    SkinThickness = st.text_input("Skin Thickness Value")

    with col2:
    Insulin = st.text_input("Insulin Level")

    with col3:
```

```
BMI = st.text_input("BMI Value")

with col1:
DiabetesPedigreeFunction = st.text_input("Diabetes Pedigree Function Value")

with col2:
Age = st.text_input("Age of the Person")



diab_diagnosis = ""

if st.button("Diabetes Test Result"):
input_data = np.array([[Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age]], dtype=float)

scaled_input_data = diabetes_scaler.transform(input_data)

diab_prediction = diabetes_model.predict(scaled_input_data)

if (diab_prediction[0] == 1):
diab_diagnosis = 'The Person is diabetic'
else:
diab_diagnosis = 'The Person is not diabetic'

st.success(diab_diagnosis)
```

```
if (selected == 'Parkisons Prediction'):
        st.title("Parkisons Prediction using ML")

        col1, col2, col3, col4, col5 = st.columns(5)

        with col1:
        MDVP_Fo_Hz = st.text_input("MDVP:Fo(Hz)")

        with col2:
        MDVP_Fhi_Hz = st.text_input("MDVP:Fhi(Hz)")

        with col3:
        MDVP_Flo_Hz = st.text_input("MDVP:Flo(Hz)")

        with col4:
        MDVP_Jitter_percent = st.text_input("MDVP:Jitter(%)")

        with col5:
        MDVP_Jitter_Abs = st.text_input("MDVP:Jitter(Abs)")

        with col1:
        MDVP_RAP = st.text_input("MDVP:RAP")

        with col2:
        MDVP_PPQ = st.text_input("MDVP:PPQ")
```

```
with col3:
Jitter_DDP = st.text_input("Jitter:DDP")

with col4:
MDVP_Shimmer = st.text_input("MDVP:Shimmer")

with col5:
MDVP_Shimmer_dB = st.text_input("MDVP:Shimmer(dB)")

with col1:
Shimmer_APQ3 = st.text_input("Shimmer:APQ3")

with col2:
Shimmer_APQ5 = st.text_input("Shimmer:APQ5")

with col3:
MDVP_APQ = st.text_input("MDVP:APQ")

with col4:
Shimmer_DDA = st.text_input("Shimmer:DDA")

with col5:
NHR = st.text_input("NHR")

with col1:
HNR = st.text_input("HNR")

with col2:
```

```
RPDE = st.text_input("RPDE")

with col3:
DFA = st.text_input("DFA")

with col4:
spread1 = st.text_input("spread1")

with col5:
spread2 = st.text_input("spread2")

with col1:
D2 = st.text_input("D2")

with col2:
PPE = st.text_input("PPE")



park_diagnosis = ""

if st.button("Parkinson's Test Result"):
input_data = np.array([[MDVP_Fo_Hz, MDVP_Fhi_Hz, MDVP_Flo_Hz, MDVP_Jitter_percent, MDVP_Jitter_Abs, MDVP_RAP, MDVP_PPQ, Jitter_DDP, MDVP_Shimmer, MDVP_Shimmer_dB, Shimmer_APQ3, Shimmer_APQ5, MDVP_APQ, Shimmer_DDA, NHR, HNR, RPDE, DFA, spread1, spread2, D2, PPE]], dtype=float)

scaled_input_data = parkinsons_scaler.transform(input_data)
```

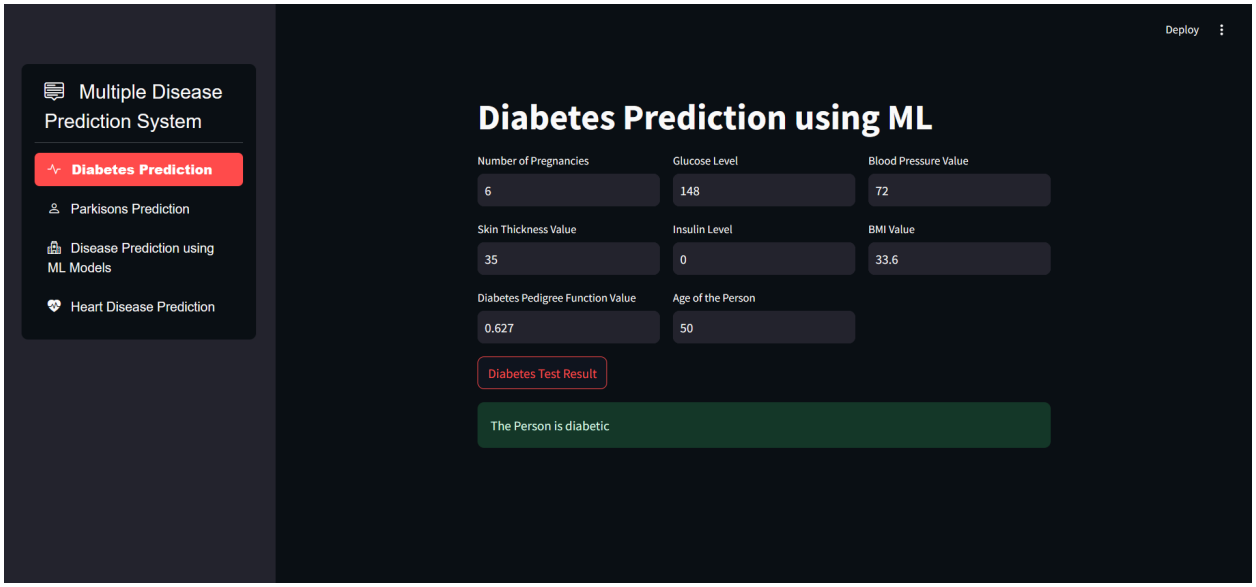
```
        park_prediction = parkinsons_model.predict(scaled_input_data)

        if (park_prediction[0] == 1):
        park_diagnosis = 'The Person is having Parkinsons'
        else:
        park_diagnosis = 'The Person is not having Parkinsons'

        st.success(park_diagnosis)


if selected == "Disease Prediction using ML Models":
        st.title("Disease Prediction by Symptoms")

        symptoms_input = []

        for i in range(5):  # Loop to show 5 symptoms dropdowns
        symptom = st.selectbox(f"Choose symptom {i+1}", l1)
        symptoms_input.append(symptom)

        symptoms_input_binary = [1 if symptom in symptoms_input else 0 for symptom in l1]

        symptoms_input_array = np.array([symptoms_input_binary]).astype(int)

        if st.button("Prediction 1"):
        prediction_dt = decision_tree_model.predict(symptoms_input_array)
        disease_name = disease_map.get(prediction_dt[0], "Unknown disease")
        st.success(f"Predicted Disease: {disease_name}")
```

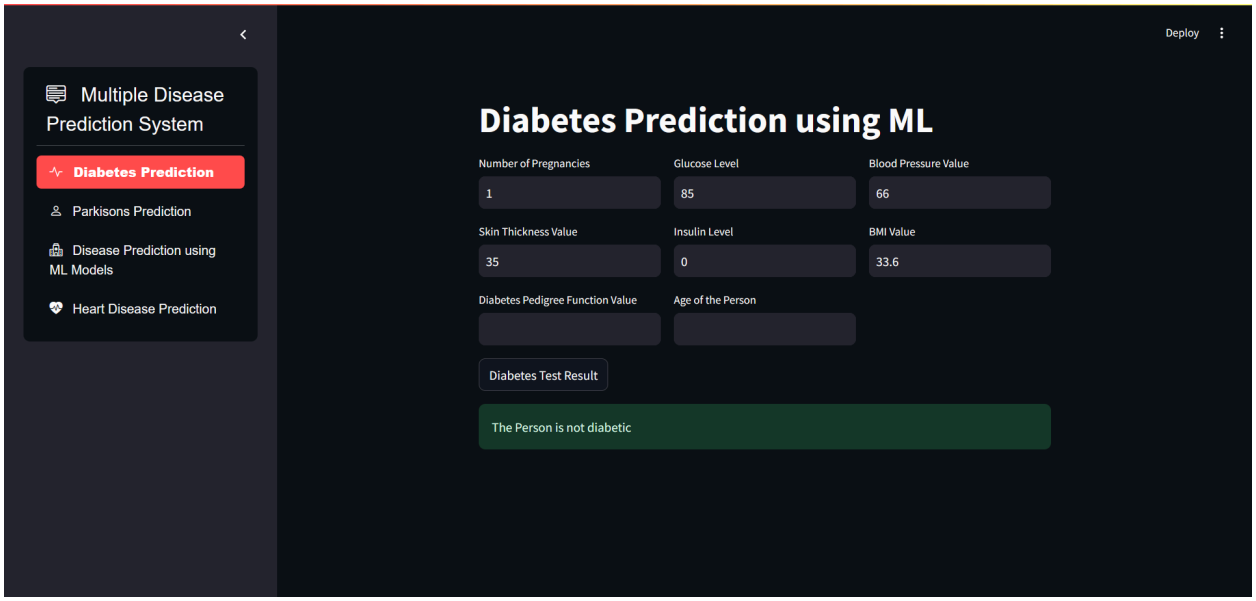
```
        if st.button("Prediction 2"):
        prediction_rf = Random_forest_model.predict(symptoms_input_array)
        disease_name = disease_map.get(prediction_rf[0], "Unknown disease")
        st.success(f"Predicted Disease: {disease_name}")

        if st.button("Prediction 3"):
        prediction_nb = naive_bayes_model.predict(symptoms_input_array)
        disease_name = disease_map.get(prediction_nb[0], "Unknown disease")
        st.success(f"Predicted Disease: {disease_name}")



if selected == "Cardiovascular Disorder Prediction":

        st.title("Cardiovascular Disorder Prediction using Stacking Model")

        col1, col2, col3 = st.columns(3)

        with col1:
        Age = st.text_input("Age")
        with col2:
        Sex = st.selectbox("Sex", ['M', 'F'])
        with col3:
        ChestPainType = st.selectbox("ChestPainType", ['TA', 'ATA', 'NAP', 'ASY'])

        with col1:
        RestingBP = st.text_input("Resting Blood Pressure")
        with col2:
```

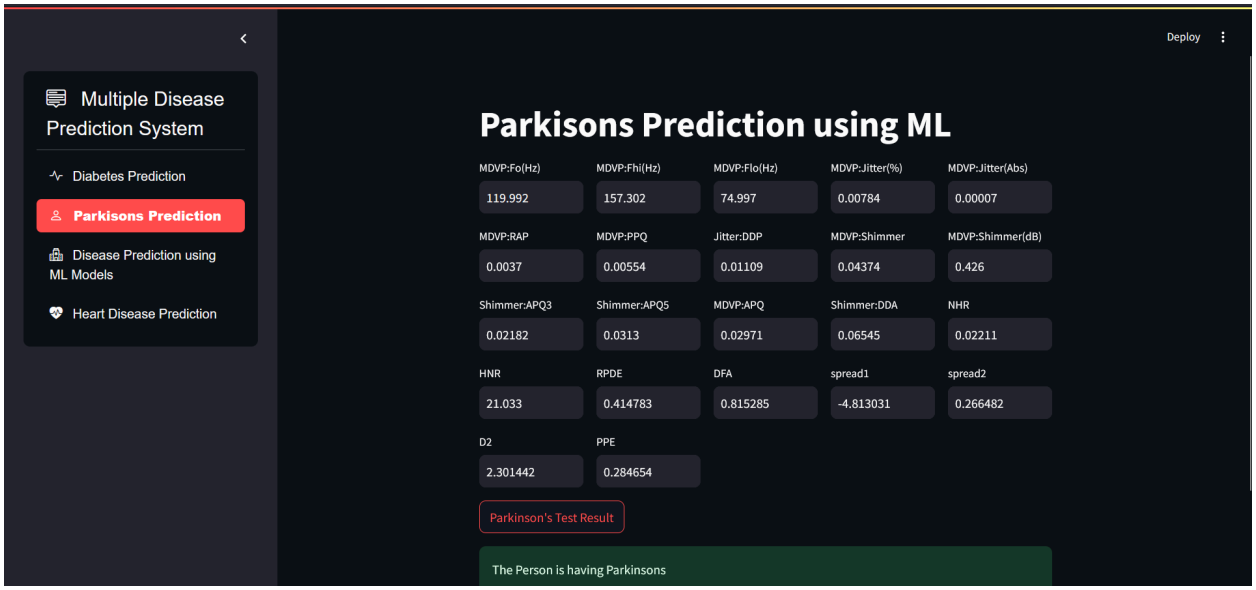
```
Cholesterol = st.text_input("Cholesterol")
with col3:
FastingBS = st.selectbox("Fasting Blood Sugar > 120 mg/dl", [1, 0])

with col1:
RestingECG = st.selectbox("RestingECG", ['Normal', 'ST', 'LVH'])
with col2:
MaxHR = st.text_input("Max Heart Rate")
with col3:
ExerciseAngina = st.selectbox("ExerciseAngina", ['Y', 'N'])

with col1:
Oldpeak = st.text_input("Oldpeak (Depression)")
with col2:
ST_Slope = st.selectbox("ST_Slope", ['Up', 'Flat', 'Down'])


if st.button("Cardiovascular Disorder Test Result"):

input_data = {
'Age': [Age],
'Sex': [Sex],
'ChestPainType': [ChestPainType],
'RestingBP': [RestingBP],
'Cholesterol': [Cholesterol],
'FastingBS': [FastingBS],
'RestingECG': [RestingECG],
'MaxHR': [MaxHR],
```

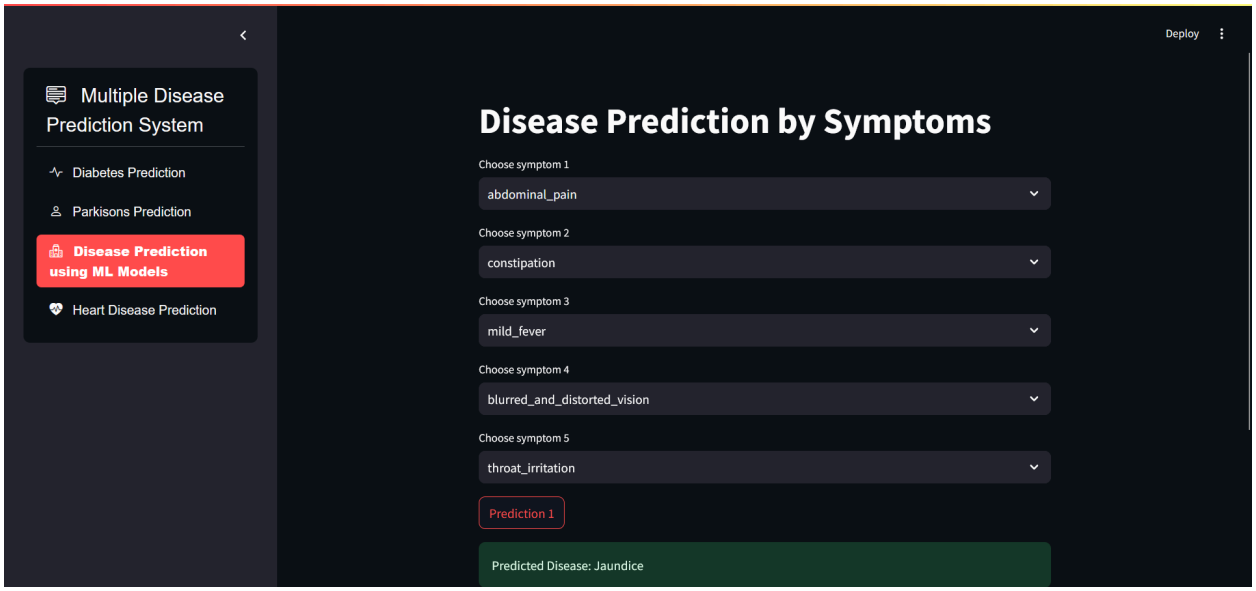
```
        'ExerciseAngina': [ExerciseAngina],
        'Oldpeak': [Oldpeak],
        'ST_Slope': [ST_Slope]
        }

        input_df = pd.DataFrame(input_data)

        categorical_columns = ['Sex', 'ChestPainType', 'RestingECG', 'ExerciseAngina',
'ST_Slope']
        for col in categorical_columns:
        input_df[col] = label_encoders[col].transform(input_df[col])

        numerical_columns = ['Age', 'RestingBP', 'Cholesterol', 'MaxHR', 'Oldpeak']
        input_df[numerical_columns] =
scaler.transform(input_df[numerical_columns].astype(float))

        expected_order = ['Age', 'Sex', 'ChestPainType', 'RestingBP', 'Cholesterol',
                'FastingBS', 'RestingECG', 'MaxHR', 'ExerciseAngina', 'Oldpeak', 'ST_Slope']

        input_df = input_df[expected_order]

        prediction = stacking_model.predict(input_df)

        if prediction[0] == 1:
        st.success("The Person is likely to have Cardiovascular Disorder")
        else:
        st.success("The Person is unlikely to have Cardiovascular Disorder")
```

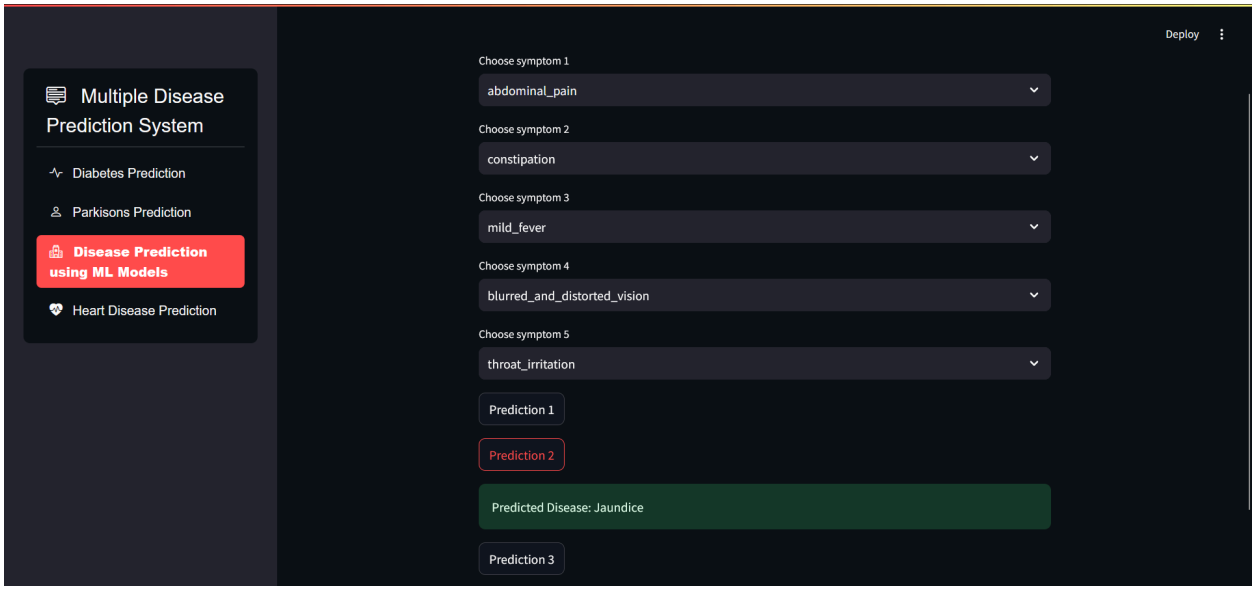
UI Output:

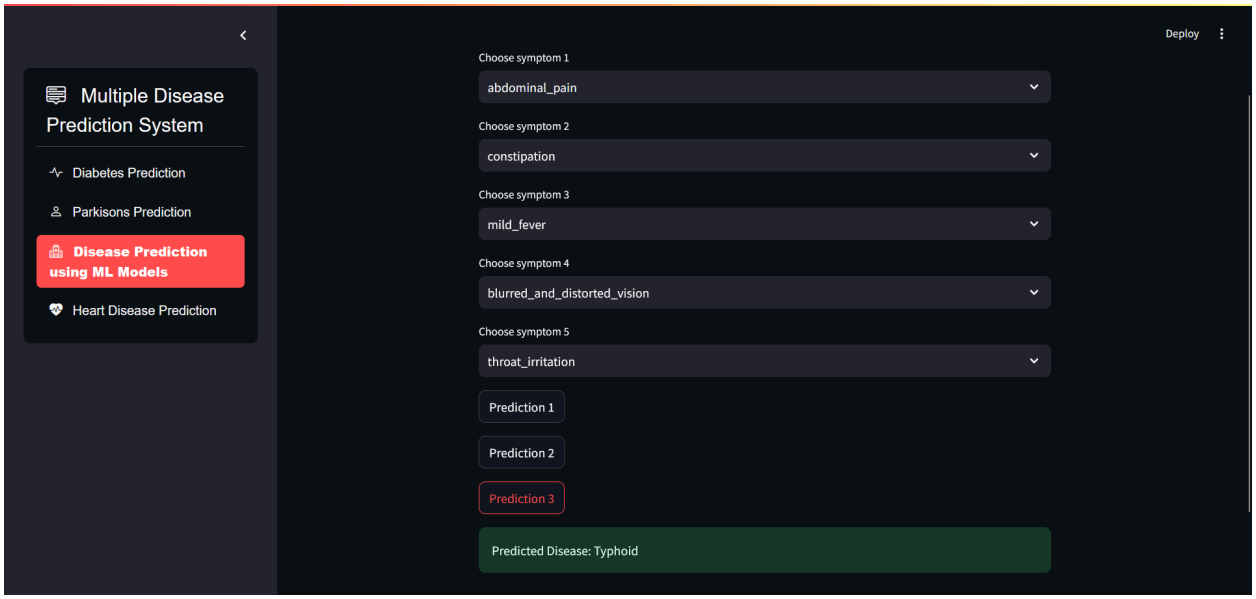

Deploy
Multiple Disease Prediction System
Diabetes Prediction
Parkisons Prediction
Disease Prediction using ML Models
Heart Disease Prediction
Diabetes Prediction using ML
Number of Pregnancies
6
Glucose Level
148
Blood Pressure Value
72
Skin Thickness Value
35
Insulin Level
0
BMI Value
33.6
Diabetes Pedigree Function Value
0.627
Age of the Person
50
Diabetes Test Result
The Person is diabetic

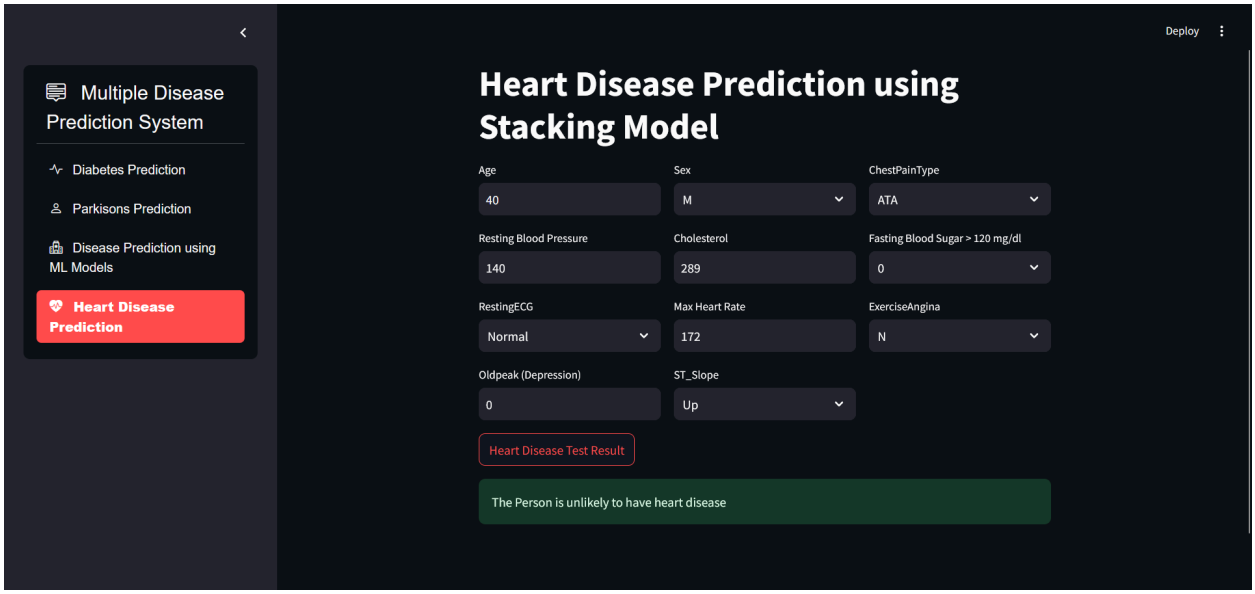

Deploy
Multiple Disease Prediction System
Diabetes Prediction
Parkisons Prediction
Disease Prediction using ML Models
Heart Disease Prediction
Diabetes Prediction using ML
Number of Pregnancies
1
Glucose Level
85
Blood Pressure Value
66
Skin Thickness Value
35
Insulin Level
0
BMI Value
33.6
Diabetes Pedigree Function Value
Age of the Person
Diabetes Test Result
The Person is not diabetic


Multiple Disease Prediction System
Diabetes Prediction
Parkisons Prediction
Disease Prediction using ML Models
Heart Disease Prediction
Deploy
Parkisons Prediction using ML
MDVP:Fo(Hz)
119.992
MDVP:Fhi(Hz)
157.302
MDVP:Flo(Hz)
74.997
MDVP:Jitter(%)
0.00784
MDVP:Jitter(Abs)
0.00007
MDVP:RAP
0.0037
MDVP:PPQ
0.00554
Jitter:DDP
0.01109
MDVP:Shimmer
0.04374
MDVP:Shimmer(dB)
0.426
Shimmer:APQ3
0.02182
Shimmer:APQ5
0.0313
MDVP:APQ
0.02971
Shimmer:DDA
0.06545
NHR
0.02211
HNR
21.033
RPDE
0.414783
DFA
0.815285
spread1
-4.813031
spread2
0.266482
D2
2.301442
PPE
0.284654
Parkinson's Test Result
The Person is having Parkinsons


Multiple Disease Prediction System
Diabetes Prediction
Parkisons Prediction
Disease Prediction using ML Models
Heart Disease Prediction
Deploy
Disease Prediction by Symptoms
Choose symptom 1
abdominal_pain
Choose symptom 2
constipation
Choose symptom 3
mild_fever
Choose symptom 4
blurred_and_distorted_vision
Choose symptom 5
throat_irritation
Prediction 1
Predicted Disease: Jaundice


Deploy
Multiple Disease Prediction System
Diabetes Prediction
Parkisons Prediction
Disease Prediction using ML Models
Heart Disease Prediction
Choose symptom 1
abdominal_pain
Choose symptom 2
constipation
Choose symptom 3
mild_fever
Choose symptom 4
blurred_and_distorted_vision
Choose symptom 5
throat_irritation
Prediction 1
Prediction 2
Predicted Disease: Jaundice
Prediction 3


Deploy
Multiple Disease Prediction System
Diabetes Prediction
Parkisons Prediction
Disease Prediction using ML Models
Heart Disease Prediction
Choose symptom 1
abdominal_pain
Choose symptom 2
constipation
Choose symptom 3
mild_fever
Choose symptom 4
blurred_and_distorted_vision
Choose symptom 5
throat_irritation
Prediction 1
Prediction 2
Prediction 3
Predicted Disease: Typhoid


Multiple Disease Prediction System
Diabetes Prediction
Parkisons Prediction
Disease Prediction using ML Models
Heart Disease Prediction
Deploy
Heart Disease Prediction using Stacking Model
Age
40
Sex
M
ChestPainType
ATA
Resting Blood Pressure
140
Cholesterol
289
Fasting Blood Sugar > 120 mg/dl
0
RestingECG
Normal
Max Heart Rate
172
ExerciseAngina
N
Oldpeak (Depression)
0
ST_Slope
Up
Heart Disease Test Result
The Person is unlikely to have heart disease

Chapter7: Key Features

- **Multiple Disease Support:** The system predicts 43 diseases, each with specific symptoms, covering common infections, chronic illnesses, skin conditions, as well as complex disorders. It is adaptable for consumers looking for health insights because it covers a broad range of medical disorders. Numerical encoding of disease labels facilitates effective model training and prediction.
- **Fast Prediction:** Real-time prediction capabilities are offered by machine learning models such as Decision Tree, Random Forest, and Naive Bayes, which can evaluate user symptoms in a matter of seconds. Preloading models increase efficiency, which makes it appropriate for settings with limited computing resources or time constraints.
- **Lightweight as well as Resource Efficient:** In addition to using effective machine learning models, the system is made for typical local computers and doesn't require a powerful GPU or cloud computing infrastructure. For web deployment, its lightweight design enables smooth interaction with Python-based frameworks such as Streamlit.
- **Expandable as well as Modular Design:** Because the system is flexible, developers may change or add models or illnesses as needed. It is future-proof and scalable for changing medical datasets and prediction requirements since it can easily integrate new symptoms, disease classifications, and machine learning techniques.

Chapter 8: Limitations

- **Not a Substitute for Professional Medical Advice:** The predictions made by the algorithm are based on statistical patterns rather than professional judgment and take into account actual medical diagnoses. It functions as a support tool rather than a diagnostic authority and shouldn't be used in place of speaking with a licensed healthcare professional.
- **Predictions Depend Heavily on Input Quality:** Because the accuracy of the system depends on user-inputted symptoms, it is extremely vulnerable to human error during data entry, as inaccurate inputs may result in inaccurate predictions.
- **Lack of Real-Time Learning or Adaptation:** Without frequent retraining, the existing implementation's static, pre-trained models may not adjust over time, which might result in performance degradation or a failure to reflect current illness patterns.
- **Privacy as well as Ethical Concerns:** Because of insufficient data governance and encryption, users may inadvertently divulge private health information using a tool, posing ethical and data privacy issues.

Chapter 9: Conclusion as well as Future Scope

Conclusion:

The project uses Python's Streamlit framework and machine learning to create an easy-to-use disease prediction system. Based on symptoms entered by the user, it predicts more than 40 diseases using models such as Decision Tree, Random Forest, and Naive Bayes. Accessibility and real-time forecasts without requiring a lot of processing power are guaranteed by the web-based interface. It is a useful first diagnostic tool in distant or resource-constrained environments, but it should not be used in place of expert medical guidance.

Future Scope:

- **Cloud Deployment for Broader Accessibility:** Disease prediction systems hosted on the cloud by AWS, Azure, or Google Cloud provide scalability, load balancing, and simpler model changes in addition to worldwide accessibility and the removal of local installations. Additionally, it permits real-time patient data access through interface with centralized health databases or APIs.
- **Integration with IoT Health Monitoring Devices:** Smartwatches, glucose monitors, pulse oximeters, wearable ECG sensors, and other Internet of Things-enabled medical equipment can be connected to the system to provide real-time physiological data analysis, continuous health monitoring, early warning alarms, and enhanced data accuracy for individualized diagnosis.
- **Expansion to Include Additional Diseases as well as Advanced Deep Learning Models:** Although it can be expanded to encompass rare or difficult diseases, the current system supports about 43 disorders. Prediction accuracy can be increased by utilizing deep learning models like neural networks and by incorporating more extensive medical datasets. Furthermore, both transfer learning and ensemble learning can improve the generalizability and robustness of the model.
- **Enhanced Data Has well asling for Improved Model Performance:** For better model results, it is essential to increase the number, diversity, and quality of training data. Context awareness is improved by including both structured and unstructured data. The

training process is strengthened by methods including anomaly identification, data augmentation, and synthetic data production. Improving preprocessing pipelines increases both the accuracy and efficiency of the model.

References

https://www.sciencedirect.com/topics/engineering/data-preprocessing

https://docs.streamlit.io/

https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://www.kipi.ai/insights/building-a-streamlit-application-with-custom-components-a-step-by-step-guide/&ved=2ahUKEwjs---JtoKNAxX5yDgGHXdIJ3sQFnoECAUQAw&usg=AOvVaw2RP_mq7MhzcKDOxHqRX3AR

https://docs.python.org/3/library/pickle.html

https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction

https://www.kaggle.com/datasets/vikasukani/parkinsons-disease-data-set

https://www.kaggle.com/datasets/kaushil268/disease-prediction-using-machine-learning

https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database

https://scikit-learn.org/

https://archive.ics.uci.edu/