

HiLCoE

School of Computer Science and Technology

**Department of Software Engineering**

**Artificial Intelligence**

**(SE434)**

**Diseases Diagnostic System Model Training Documentation**

|  |  |
| --- | --- |
| **Name** | **ID** |
| Rami Nabil | KM9058 |
| Kalkidan Tesfaye | KY0326 |
| Assier Anteneh | WQ0129 |
| Kunuz Mohammed | GX3230 |
| Yabsira Kassaye | YC3901 |
| Surafeal Fasil | XH7610 |

Table of Contents

[1. Introduction I](#_Toc186646759)

[1.1. Overview I](#_Toc186646760)

[1.2. Purpose and Objectives I](#_Toc186646761)

[1.3. Supported Models I](#_Toc186646762)

[2. Workflow Overview I](#_Toc186646763)

[2.1. Common Pipeline I](#_Toc186646764)

[2.2. Model-Specific Pipelines II](#_Toc186646765)

[3. Prerequisites II](#_Toc186646766)

[3.1 Software Requirements II](#_Toc186646767)

[3.2 Dataset Descriptions III](#_Toc186646768)

[4. Detailed Process III](#_Toc186646769)

[4.1 Data Loading and Exploration III](#_Toc186646770)

[4.2 Preprocessing and Scaling III](#_Toc186646771)

[4.3 Model Training and Tuning III](#_Toc186646772)

[4.4 Model Evaluation III](#_Toc186646773)

[4.5 Serialization IV](#_Toc186646774)

[5. Model-Specific Features and Insights IV](#_Toc186646775)

[5.1 Diabetes Model IV](#_Toc186646776)

[5.2 Heart Disease Model IV](#_Toc186646777)

[5.3 Parkinson’s Disease Model IV](#_Toc186646778)

[6. Support Vector Machines (SVM) IV](#_Toc186646779)

[6.1. How Does SVM Work? V](#_Toc186646780)

[6.2. Mathematical Representation VI](#_Toc186646781)

[6.3. Advantages of SVM VI](#_Toc186646782)

[6.4. Limitations of SVM VII](#_Toc186646783)

[6.5. Relevance of SVM in the Script VII](#_Toc186646784)

[7. Deployment and Usage VIII](#_Toc186646785)

[8. Error Handling VIII](#_Toc186646786)

[9. Conclusion IX](#_Toc186646787)

# Introduction

## Overview

The provided script is a modular machine learning pipeline designed to predict three critical diseases using Support Vector Machines (SVM). These models cater to binary classification problems in the healthcare domain.

## Purpose and Objectives

• **Purpose**: Provide a robust, unified framework for disease prediction.

• **Objectives**:

• Enhance early detection of diseases for timely medical intervention.

• Facilitate a scalable and modular approach for additional diseases.

• Maintain high accuracy and generalizability across datasets.

## 1.3. Supported Models

1. **Diabetes Prediction**: Detects diabetes risk based on patient health metrics.

2. **Heart Disease Prediction**: Assesses cardiovascular health risks.

3. **Parkinson’s Disease Prediction**: Diagnoses Parkinson’s disease using vocal biometrics.

# 2. Workflow Overview

## 2.1. Common Pipeline

The script follows a unified workflow applicable to all three models:

1. **Data Loading**: Import datasets using Pandas.

2. **Preprocessing**:

• Handle missing values.

• Standardize features.

3. **Model Training**:

• Train an SVM classifier on 80% of the data.

4. **Evaluation**:

• Test the model on the remaining 20% of the data.

• Compute training and test accuracies.

5. **Serialization**:

• Save the trained model and scaler for deployment.

## 2.2. Model-Specific Pipelines

Each model has distinct preprocessing requirements and features:

• **Diabetes**: Uses health metrics like glucose levels and BMI.

• **Heart Disease**: Focuses on cardiovascular and lifestyle factors.

• **Parkinson’s**: Relies on voice biometrics, such as jitter and shimmer.

# 3. Prerequisites

## 3.1 Software Requirements

• Python 3.6 or higher

• Required libraries:

• numpy: For numerical operations.

• pandas: For data manipulation.

• scikit-learn: For machine learning algorithms.

• pickle: For model serialization.

## 3.2 Dataset Descriptions

• **Diabetes Dataset**: Metrics such as glucose levels, insulin levels, and BMI.

• **Heart Disease Dataset**: Features like cholesterol, chest pain type, and resting ECG.

• **Parkinson’s Dataset**: Vocal features, excluding identifiers like name.

# 4. Detailed Process

## 4.1 Data Loading and Exploration

• Load datasets using pd.read\_csv.

• Inspect the data using .info(), .describe(), and .isnull().sum().

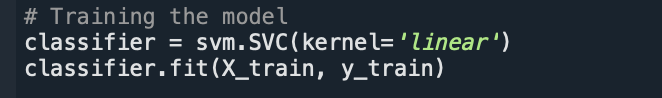
## 4.2 Preprocessing and Scaling

1. Drop irrelevant columns.

2. Use StandardScaler to standardize feature values for SVM compatibility.

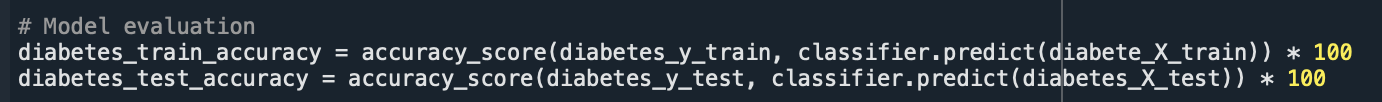
## 4.3 Model Training and Tuning

• Split data into training and testing sets using train\_test\_split.

 • Train the SVM model with:

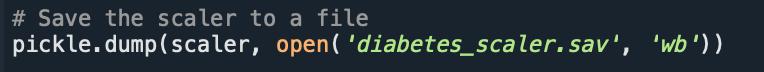
## 4.4 Model Evaluation

Evaluate the trained model with:



## 4.5 Serialization

Save models and scaler for deployment:





# 5. Model-Specific Features and Insights

## 5.1 Diabetes Model

• **Key Features**: Glucose, BMI, Insulin.

• **Insights**: High glucose and BMI values strongly indicate diabetes.

## 5.2 Heart Disease Model

• **Key Features**: Cholesterol, chest pain, ECG results.

• **Insights**: Combining ECG results with cholesterol metrics improves predictions.

## 5.3 Parkinson’s Disease Model

• **Key Features**: Jitter, Shimmer, HNR.

• **Insights**: Vocal variations correlate strongly with Parkinson’s symptoms.

# 6. Support Vector Machines (SVM)

Support Vector Machine (SVM) is a supervised machine learning algorithm primarily used for classification and regression tasks. SVM is especially effective for high-dimensional datasets and binary classification problems.

* **Key Objective**: SVM aims to find the optimal hyperplane that separates data points of different classes in the feature space.
* **Applications**: Widely used in applications like text classification, image recognition, and medical diagnosis.

## 6.1. How Does SVM Work?

SVM constructs a decision boundary (or hyperplane) that maximizes the margin between data points of different classes:

1. **Feature Space**: Each input data point is represented as a vector in a high-dimensional space.

* For example, in a 2D space, the features are represented as points on a plane.

2. **Hyperplane**: A hyperplane is a linear decision boundary that separates classes.

* For a 2D dataset, the hyperplane is a line. For a 3D dataset, it’s a plane.

3. **Maximizing the Margin**: SVM selects the hyperplane with the maximum margin, i.e., the largest distance between the hyperplane and the nearest data points from each class (called **support vectors**).

* A large margin reduces the risk of misclassification and improves generalization.

4. **Kernel Trick**: For non-linearly separable data, SVM uses the **kernel trick** to transform data into a higher-dimensional space where a linear hyperplane can separate the classes. Common kernels include:

* **Linear Kernel**: Suitable for linearly separable data.
* **Polynomial Kernel**: Captures polynomial relationships between features.
* **Radial Basis Function (RBF) Kernel**: Captures non-linear patterns.

5. **Support Vectors**: Only the data points closest to the hyperplane (support vectors) are used to define the decision boundary. These points are critical for training the SVM.

## 6.2. Mathematical Representation

For a binary classification problem, the hyperplane is defined by:

WTx + b = 0

Where:

• w : Weight vector perpendicular to the hyperplane.

• x : Feature vector.

• b : Bias term.

The optimization problem for maximizing the margin is:

½||w||2

Subject to: yi(wTxi + b)1 for all i

Where:

• yi is the class label (+1 or -1).

• xi  is the data point.

## 6.3. Advantages of SVM

1. **Effective in High-Dimensional Spaces**:

* SVM performs well even when the number of features exceeds the number of samples.

1. **Robust to Overfitting**:

* The margin maximization principle reduces the likelihood of overfitting.

1. **Versatile Kernels**:

* The kernel trick allows SVM to handle both linear and non-linear classification tasks.

1. **Works Well for Binary Classification**:

* SVM is particularly strong for binary problems, like those in this script.

## 6.4. Limitations of SVM

1. **Computational Complexity**:

* Training SVM can be slow for very large datasets due to quadratic optimization.

1. **Sensitivity to Hyperparameters**:

* Requires careful tuning of parameters like the regularization parameter  C  and kernel parameters.

1. **Less Effective with Noise**:

* Sensitive to overlapping classes and outliers.

## 6.5. Relevance of SVM in the Script

1. **Choice of SVM**:

* SVM’s strength in handling high-dimensional data makes it ideal for the datasets in this project, such as vocal metrics in the Parkinson’s dataset or health indicators in the diabetes dataset.

1. **Linear Kernel**:

* The script employs the **linear kernel** because:
* It is computationally efficient.
* The datasets in this project are structured and well-suited for linear separation.

1. **Feature Scaling**:

* SVM relies on distance metrics for hyperplane construction. To ensure features contribute equally, the script applies **standard scaling**.

1. **Binary Classification**:

* Each model deals with a binary outcome:
* Diabetes: Diabetic (1) or Non-Diabetic (0).
* Heart Disease: Has Disease (1) or No Disease (0).
* Parkinson’s: Parkinson’s (1) or No Parkinson’s (0).

1. **Generalization**:

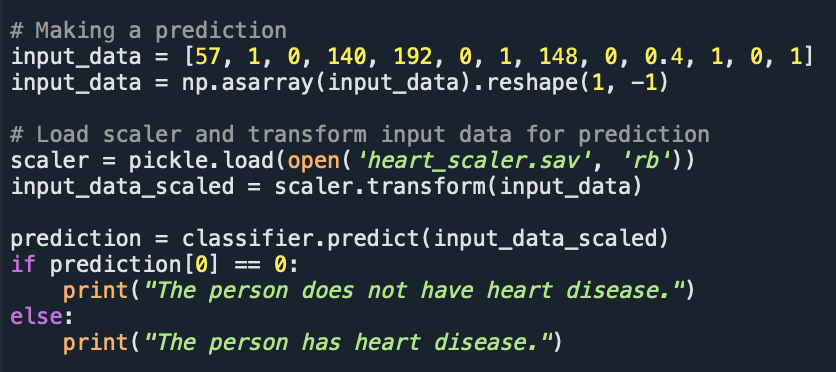
* SVM generalizes well to unseen data, as evident from the training and test accuracy reported in the script.

# 7. Deployment and Usage

**Model Loading**



**Making Predictions**



# 8. Error Handling

**Common Issues**

|  |  |  |
| --- | --- | --- |
| **Error** | **Cause** | **Solution** |
| FileNotFoundError | Missing model/scaler file. | Verify file paths. |
| ValueError | Incorrect input dimensions. | Validate input shapes. |
| ImportError | Missing library. | Install via pip install. |

# 9. Conclusion

The provided script represents a significant step toward integrating machine learning into healthcare diagnostics. By leveraging Support Vector Machines (SVM), it demonstrates the potential for accurate and reliable predictions in detecting diabetes, heart disease, and Parkinson’s disease. This solution not only aids healthcare professionals in decision-making but also empowers patients to monitor their health proactively.

Beyond its immediate applications, the script is designed with scalability and flexibility in mind. It can be extended to support additional diseases or incorporate more sophisticated models, such as deep learning, as datasets and computational resources grow. Furthermore, it aligns well with the broader trends in telemedicine and wearable health technology, where such predictive tools can be seamlessly integrated for real-time monitoring and diagnosis.

The methodology of standardization, model evaluation, and serialization ensures the framework remains robust, reproducible, and easy to deploy in production environments. Its modular design allows for integration into larger systems, such as hospital information systems or cloud-based diagnostic platforms, making it a valuable asset in the journey toward smarter healthcare solutions.

This project underscores the importance of bridging technology and medicine to address some of the most pressing health challenges. As the healthcare industry continues to evolve, the role of machine learning in improving patient outcomes and reducing healthcare costs will become increasingly critical. By building on this foundation, developers, researchers, and healthcare practitioners can work together to deliver innovative solutions that transform lives.