Software Requirements Specification

for

ML Library

Version 1.0 approved

Prepared by Nilay Kamat(IMT2021096)
Kalyan Ram(IMT2021023)
Madhav Sood(IMT2021009)
Shlok Agrawal(IMT2021103)

IIIT Bangalore

26th October, 2023

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

The purpose of this SRS document is to outline the requirements and specifications for the development of a Machine Learning Library that includes two primary models: the Naive-Bayes Classifier and the K-Nearest Neighbors (KNN) algorithm. This library aims to provide a comprehensive set of tools for data scientists and developers to create and evaluate classification models.

1.2 Intended Audience and Reading Suggestions

The intended audience includes software developers, data scientists, machine learning engineers, and project managers involved in the development, testing and use of the Machine Learning Library. This document will help in understanding the features, functionalities, and constraints of the library.

1.3 Product Scope

The Machine Learning Library is designed to be a versatile tool that includes the implementation of the Naive-Bayes Classifier and K-Nearest Neighbors algorithms. It enables data preprocessing, model building, evaluation, and integration into applications. The library aims to provide robust, efficient, and easy-to-use machine learning capabilities.

2. Overall Description

2.1 Product Perspective

The Machine Learning Library is a standalone product that can be used independently or integrated into various applications. It does not rely on external systems or databases. It is designed to work with diverse datasets and provides a high degree of flexibility for model development.

2.2 Product Functions

The main features of the Machine Learning Library include:

1. Data Preprocessing: Capabilities for data loading, cleaning, and transformation.
2. Naive-Bayes Classifier: Implementation of the Naive-Bayes classification algorithm.
3. K-Nearest Neighbors (KNN): Implementation of the KNN classification algorithm.
4. Model Training and Prediction: Training either of the models on the given dataset and using them for prediction.
5. Model Evaluation: Tools for model evaluation, including metrics and cross-validation.
6. User-Friendly API: A user-friendly API for model creation and evaluation.
7. Scalability: Ability to handle large datasets and efficient model training.

2.3 Operating Environment

The operating environment for the Machine Learning Library, developed in Java, is designed to be versatile and compatible with multiple operating systems. It includes the following key points:

- **Operating Systems:** The library is compatible with Windows, Linux, and macOS, ensuring broad OS support.
- **Java Runtime Environment (JRE):** Users need Java SE 8 or higher to run the library.

3. External Interface Requirements

3.1 User Interface

The primary user interface for data scientists and developers is a Python API. This API includes a variety of classes and functions for data preprocessing, model creation, training, evaluation, and model integration. It is designed to be user-friendly and well-documented, enabling users to efficiently work with the library.

3.2 Hardware Interfaces

The Machine Learning Library is designed to not rely on specific hardware interfaces. It is expected to run on standard computing hardware and does not require specialized hardware components.

3.3 Software Interfaces

The software interfaces for the library are:

1. The library must interface with data sources, such as databases, data lakes, and APIs.
2. It should be compatible with popular ML frameworks and libraries.

4. System Features

4.1 Data Preprocessing

Description: Data preprocessing is a critical feature for ensuring data quality before model training.

Priority: High priority due to its fundamental role in model accuracy.

Functional Requirements:

1. Data loading from various sources.
2. Data cleaning, including handling missing values.
3. Data transformation, scaling, and encoding.
4. Feature engineering for creating new features.
5. Error handling for robustness.
6. Data preprocessing configuration management.

4.2 Naive-Bayes Classifier Implementation

Description: This feature involves implementing the Naive-Bayes classification algorithm.

Priority: Medium priority for text classification and other applications.

Functional Requirements:

1. Training with labeled data.
2. Classification of new data points.
3. Model evaluation with metrics.
4. Model persistence for reusability.
5. Customization options for Laplace smoothing, class priors, etc.

4.3 K-Nearest Neighbors (KNN) Implementation

Description: This feature involves implementing the K-Nearest Neighbors (KNN) classification algorithm.

Priority: Medium priority for its versatility.

Functional Requirements:

1. Training with labeled data, distance metric, and neighbors.
2. Classification of new data points with neighbor information.
3. Model evaluation with metrics.
4. Model persistence for reusability.
5. Customization options for distance metrics and parameters.
6. These summaries provide a concise overview of each feature's purpose, priority, and main functional requirements.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

1. **Efficiency**: The library provides efficient model training and inference.
2. **Scalability**: It scales to handle large datasets and support distributed computing where applicable.

5.2 Safety Requirements

1. **Error Handling**: The library will provide informative error messages that help the user identify the errors in their code.
2. **Data Validation**: The library will have mechanisms to ensure that the input data for training and testing is in the desired format.

5.3 Security Requirements

The ML library won’t have any built-in security features and doesn’t expect security requirements from the user.

5.4 Software Quality Attributes

1. **Usability**: The library will be user-friendly and well-documented to ensure ease of use.

2. **Maintainability**: It is maintainable, i.e the code is well organized and documented , which makes it easier to add future updates and fix existing bugs.
3. **Reliability:** The ML library will provide the same outputs as long as the inputs and the parameters stay the same. This ensures consistency in results during training and prediction.