ATILIM ÜNİVERSİTESİ
1 9 9 6

SE 494 Senior Project Report

HUMAN ACTIVITY RECOGNITION

Team Members	Batuhan GÜNEŞ
	Bana HIDMI
	N. Oğuzhan TEMİZKAN
	M. Berk TÜZEMEN
Supervisor	Dr. Güler KALEM

ABSTRACT

In this project, it predicts 6 different human activities with three different non-linear classification models. Accuracy of 0.9915 for SVM, 0.9805 for KNN and 0.9821 for Rf. The general purpose of Bı classifications is to maximize accuracy and bring the prediction value closer to the real value. As a result, SVM with this value of 0.9915, absolute accuracy was obtained using the correct parameters. If better parameters are found and tested in future plans, the absolute value of 1 can be approached more. These parameters can be evaluated with better data sets and used in the testing phase.

Keywords: Mobile Sensors, Machine Learning, gyroscope & mobile sensors, Support Vector Classifier, classification

ACKNOWLEDGEMENTS

We would like to express our deepest appreciation to all those who provided us the possibility to complete this project. A special gratitude we give to our senior project supervisor, Dr. Güler KALEM to all her understanding, patience and help.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

HAR: Human Activity Recognition

SVM: Support Vector Machine

KNN: k-Nearest Neighbors

RF: Random Forest

CNN: Convolutional Neural Network

ACC: Accelerometer

GYR: Gyroscope

SRS: Software Requirements Specification

GUI: Graphical User Interface

RFE: Recursive feature elimination

CHAPTER 1 INTRODUCTION

Recognition of human activity is a very remarkable application area in engineering studies today. Expert systems analyze human movements using sensor data. The common ones among these sensors can be listed as wearable sensors and phone sensors. In order to obtain the best performance from these systems, first of all, the performance of machine learning algorithms is analyzed using these data. Then, expert systems are designed based on the most successful algorithm. The aim of this study is to identify the machine learning algorithm that successfully detects human activity by using mobile sensors. In the data set used in this study, there are records of 30 volunteer subjects who perform daily living activities [1].

Data obtained from a smart phone with embedded gyroscope and accelerometer Each subject with a smart phone performed six activities: “Walking”, “Walking Upstairs”, “Walking Downstairs”, “Sitting”, “Standing” and “Laying”. The performances of the models designed with K Nearest Neighbor, Random Forest, and SVM classifier algorithms within the framework of 10-fold cross validation technique were evaluated on this dataset. With this technique, training data were sent to these models 5 times as input data, and the performance of the models was analyzed on the test data. The performances of the models were compared using the mean accuracy measurement [2].

1.1 Purpose

This project aims to predict the movements of the person using the phone accurately and find the most efficient algorithm for training and testing the dataset.

1.2 Scope

The scope of this project is to make the most statistically accurate estimate by analyzing and comparing the data using the python programming language and to obtain the closest accuracy of the predicted estimate to the truth.

CHAPTER 2 LITERATURE REVIEW

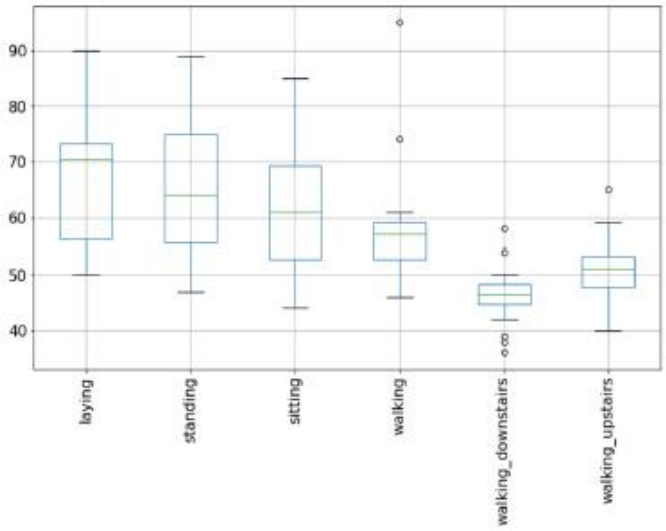
We have supported our project by reviewing literature and donating it is our own project. Selecting a suitable sensor configuration is an important aspect of recognizing human activities with wearable motion sensors. This problem encompasses selecting the number and type of the sensors, their position on the human body. In earlier studies, researchers have used customized sensor configurations, and compared them with others in terms of the activity recognition rate. However, it is clear that these comparisons are dependent on the feature sets and classifiers employed. In this study, employing mutual information measure, sensor configurations are determined with respect to the time domain distributions of the raw sensor measurements. The most informative axes such as accelerometers, gyroscopes, and magnetometers fixed at several locations on the human body are detected [2].

We can observe that there are many common points in the researches already done. Researchers use machine learning or deep learning methods to recognize human activity using mobile sensors. Since we also use machine learning in our project, we have achieved the success rates of various classifier algorithms thanks to these studies.

Since we will be working with mobile sensors, considering the location of the phone and the movements of various people may be different, the success rates of algorithms may vary in various researches. It is observed that the data set used in the research has a great effect on these success rates. Because the person who collects the data and the person who tests the algorithms are the same, it can increase the success rate. Accordingly, as a result of our research, we tried to make the most suitable algorithms for our study, by choosing the algorithms that we saw as having a high success rate in each study [3].

The mobile sensors we will use when recognizing human activity with the phone are the gyroscope and accelerometer. The gyroscope sensor is a sensor that detects the change of the spatial position of the robot. It includes a 3-axis accelerometer, a 3-axis angular velocity sensor, and a motion processor. It can detect the relative displacement changes of X-axis, Y-axis and Z-axis in space. It is used for motion tracking in self-balancing car robots, wearable devices. By using this sensor, a database can be created in which single leg use and even double leg use are detected. This method, which was developed using matrix formulas, shows which activity is in the foreground with a high rate of second differences. Classified by majority vote of its neighbors together with kNN; activity is awarded to the class most common among its nearest

neighbors. They report a classification performance of 93.07 % for SVM and 80.72 % for kNN. The data obtained as a result of the researches were compared with 5 main evaluation methods, as shown in Figure 2.1 [2].



	average	*median*	*std.deviation*
laying	66.85	70.50	11.72
standing	65.75	64.00	11.75
sitting	61.20	61.00	11.38
walking	58.05	57.00	10.43
walking down	46.20	46.50	5.06
walking upstairs	50.40	51.00	5.83

Rank	*Machine learning method*	*Accuracy*[a]
1	MultyLayerPerceptron Calassifier	0.94095
2	Support Vector Classifier	0.93077
3	Decision Tree Classifier	0.86155
4	K-Neighbors Classifier	0.80726
5	Gaussian Naïve Bayes Classifier	0.77027
6	Quadratic Discriminant Analysis Classifier	0.73566

Figure 2.1 - Classifiers Accuracy and Activity Tables

CHAPTER 3 SOFTWARE REQUIREMENTS SPECIFICATION

3.1 Introduction

This section has been written to explain the importance and purpose of requirements specification and to provide an overview of the entire content involved. The purpose of this document is to explain the details of the project, what the software will do, how the software will appear to the customer or user, the software lifecycle and development phases. In addition, this document guides the project developers in the development of the project and how the project can fully meet the needs of its users.

3.1.1 Purpose

The purpose of this document is to provide a detailed description of the software aimed at analyzing daily human activities with data obtained using mobile sensors of a mobile phone. The aim of this software is to determine the activities in the most accurate way using certain algorithms and to use them in the field of health informatics in the future. System development also explains constraints, interfaces, and other interactions. In addition, it explains functional and non-functional requirements with actors and diagrams.

3.1.2 Scope

This human activity recognition application understands 5 different human activities using mobile sensors, thanks to models trained with machine learning. Using the data collected by various participants, we analyze this data in the most accurate way with 3 different classifiers and try to maximize the accuracy rate in recognizing human activity. In general, we try to understand human activities in the most accurate way using machine learning and deep learning. The application doesn’t need the internet and works locally.

3.1.3 Overview

The remainder of this SRS document contains a description of all requirements for "Human Activity Recognition with GUI". The general description section contains general information that is not very specific and provides background for the subheadings. In this section, we also mention product perspective and functions, user classes, working environment,

constraints, assumptions, and dependencies. Here, external interface requirements, functional requirements, non-functional requirements are some of the topics.

3.2 Overall Description

3.2.1 Product Perspective

In our project, the user can analyze the success rates of the classifiers by comparing the theoretical results with the experimental results. Thanks to the application, they can get the output of their activity.

3.2.2 Product Functions

Activity classification, comparing the incoming data, showing which activity is done according to the data.

3.2.3 User Classes and Characteristics

In our application users can see the output of the activity they are currently doing, to make sure that the analysis of the data made by the three algorithms works right.

3.2.4 Operating Environment

We control our version of code with GitHub, also we use Windows 10, and the programming language is Python. We use Google Colab as a notebook because we can both work together and train our models more easily thanks to Colab GPUs.

3.2.5 Design and Implementation Constraints

- The programming language shall be Python.
- Tensorflow, Numpy, Pandas are indispensable libraries and shall be used.
- A data set with at least 30 participants shall be used.
- The Codes shall be written in a python notebook environment (file extension .ipynb).
- At least 3 classifier algorithms shall be used.
- At least 5 human activities shall be recognized and analyzed.

3.2.6 Assumptions and Dependencies

- User data constantly input in the background.
- The application shall work in the same way on all devices that have a gyroscope and an accelerometer.

3.3 External Interface Requirements

3.3.1 User Interfaces

- The program has a simple interface.
- Users can see the most accurate possible result highlighted.

3.3.2 Hardware Interfaces

- The user shall use a smart device.
- Smart devices shall have an accelerometer and a gyroscope.

3.3.3 Software Interfaces

- The system shall display 5 activities.
- The system shall display the probability of the 5 activities.

3.3.4 Communication Interfaces

- The system should display a probability table.
- The system should display the current activity result as a message.

3.4 Functional Requirements

- The system shall train the given data with 3 different algorithms.
- The system shall recognize 5 different human activities and show them to the user.
- The system should compare the success rates of different algorithms.
- The system shall support the recognition of activities and model achievements with graphs.
- The system shall give the probability rate of each activity.

3.5 Non-Functional Requirements

3.5.1 Performance Requirements

- The system shall display the activity result in 5 seconds.
- The application launch time shall take 3 seconds.
- The transition time between movements should be short and the program shall be able to make a quick detection.
- The accuracy of the final train-test result should be more than 90%.

3.5.2 Availability Requirements

- The system can run offline.
- The system shall be available and run continuously on android devices.

CHAPTER 4 DESIGN

4.1 System Overview

This document has been prepared for the software design of the "Human Activity Recognition" project. This document contains information about how the project will look and how the software will be built. Various information about the project is also given in this section.

4.1.1 Scope

The project is being developed to recognize human activities with machine learning algorithms after collecting data with phone sensors. The project includes classifier algorithms such as "k-nearest neighbors", "Random Forest" and "Support Vector Machines" from machine learning algorithms. The project recognizes human activity with mobile sensors, analyzes the data to find the success rates of the algorithms and the most successful algorithm and reaches the result. In addition, the trained models can be used in various applications to recognize human activity.

4.1.2 Purpose

The purpose of this document is to describe the software system that has been designed to meet the requirements specified in the SRS document. This project has the potential to be used in a variety of applications in the future. Also, this document explains how the project works, and how we are using different three different algorithms to achieve the highest possible accuracy in human activity recognition.

4.1.3 Intended Audience

The Software Description document is intended for the following audiences:

Developers: developers add more variety of movement models in its future applications

Users: Anyone with a smart phone can use it.

Testers: Can test for measurement errors.

4.1.4 Software Design

Machine learning model will be used in our project. KNN, SVM are software machine learning methods in which RF models are decomposed into a cluster. Thus, the system was designed from a statistical point of view. Algorithms that have already been created are examples for better and easier modeling.

4.2 Design Viewpoints

We supported our design perspective with various tables and diagrams. That will be shown in detail in the next sections.

4.2.1 Context Viewpoint

The Context view of a system defines the relationships, dependencies, and interactions between the system and its environment. Here there is a context between the user and the phone. The data received by the phone from the user is reflected back to the user.

4.2.2 Composition Viewpoint

It defines what the components that make up the compositional viewpoint do and how they can be in the right role in the defined tasks.

4.2.3 Information Viewpoint

Describes the way that the architecture stores, manipulates, manages, and distributes information.

4.2.4 Development Viewpoint

It provides information on how it was developed and supported. Development views communicate the aspects of the architecture of interest to those stakeholders involved in building, testing, maintaining, and enhancing the system.

4.2.5 Operational Viewpoint

It describes how to operate, manage, and support the system while it is running. This provides us with feedback on the plans to be made.

4.3 System and Software Architecture

The system architecture has been symbolized in a simple way and it is aimed to give an idea about the functioning of our system. As it is shown in Figure 4.1, our first step is the user, while the user is using the phone, it collects data according to the active movement of the user with the gyroscope and accelerometer sensors it has in the phone, compares it with the data set it has previously, and classifies it statistically, giving feedback to the user about which movement he/she has made.

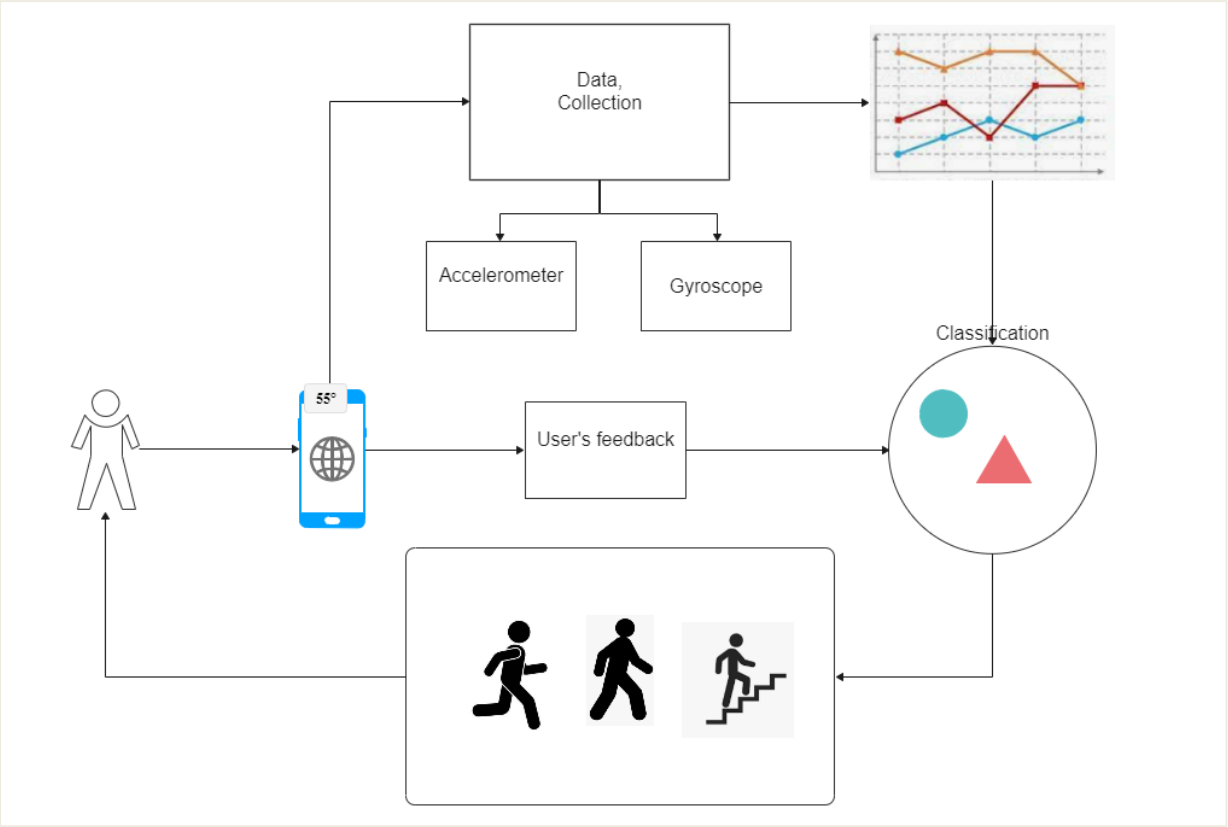


Figure 4.1 System Architecture Diagram

In the software architecture python programming language and machine learning were used together. The data is taught with machine learning and compared with different algorithms. KNN, SVM and RF are some of the algorithms we used.

4.3.1 UML Diagrams

Class diagram is one of the most frequently used diagrams of UML and aims to represent classes in object-oriented analysis, design and programming in our project in a clear and understandable way. As can be seen in Figure 4.2, the classes and functions for the various classifiers are clearly specified. Besides, some functions used for dataset are also shown in the diagram.

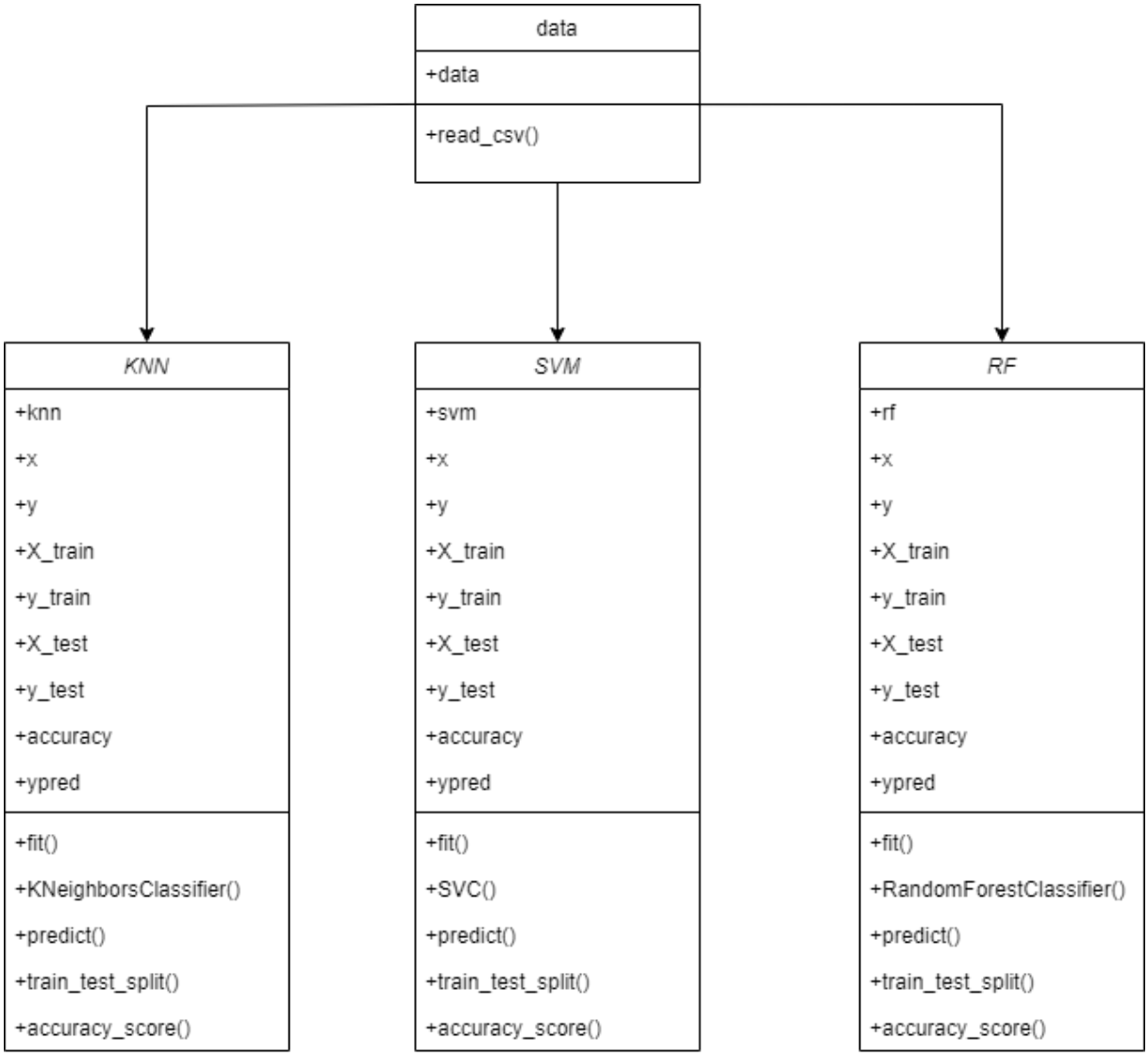


Figure 4.2 Class Diagram

The collaboration diagram tries to show the communication of the classes in our project. As seen in Figure 4.3 below, the data is processed for each classifier and the accuracy rates of these classifiers are printed on the screen and transmitted to the user by the function.

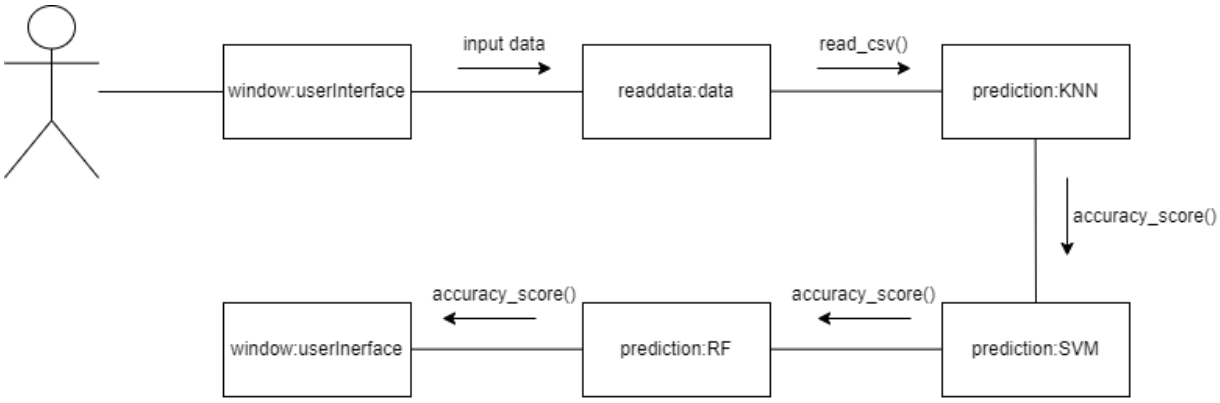


Figure 4.3 Collaboration Diagram

A deployment diagram shows the execution architecture of our project, including nodes such as hardware or software execution environments, and the middleware connecting them. As can be seen in Figure 4.4 the deployment diagram in our project that needs various files in 3 different environments and connects with 2 different nodes.

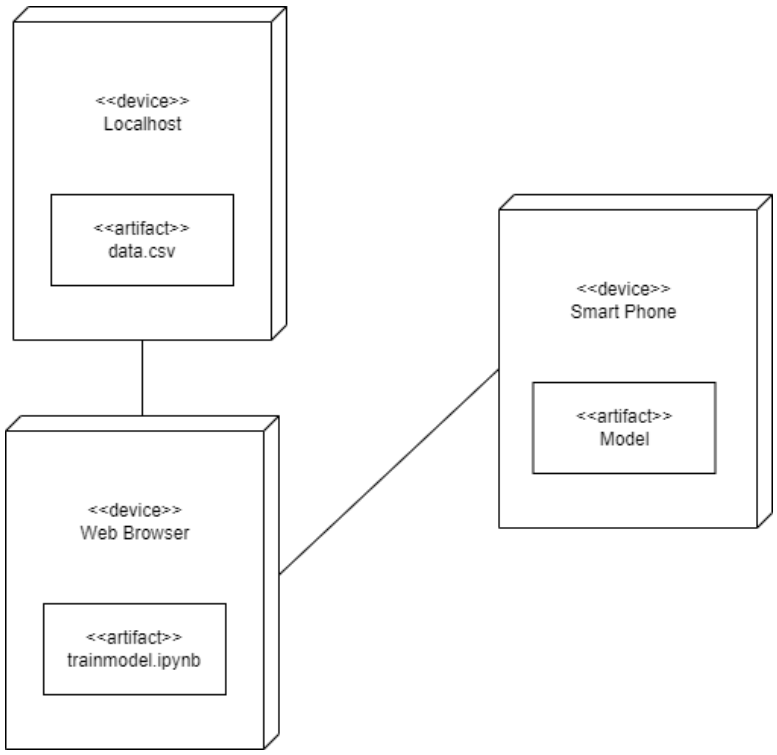


Figure 4.4 Deployment Diagram

4.4 Design of Training Scenarios

4.4.1 Dataset

We found a ready dataset to develop our project. This dataset contains some activities of some volunteer participants collected by smart phone. The trials were carried out on a group of 30 volunteers ranging in age from 19 to 48 years old. Each participant did six tasks while wearing a smart phone (Samsung Galaxy S II) around their waist (walking, walking upstairs, walking downstairs, sitting, standing, and laying). They recorded 3-axial linear acceleration and 3-axial angular velocity at a constant rate of 50Hz using the device's inbuilt accelerometer and gyroscope.

For each record in the dataset the following is provided:

- Triaxial acceleration from the accelerometer (total acceleration) and the estimated body acceleration.
- Triaxial Angular velocity from the gyroscope.
- A 561-feature vector with time and frequency domain variables.
- Its activity label.
- An identifier of the subject who carried out the experiment.

4.4.2 Training, Testing and Finding Accuracy

To avoid over fitting, we decided to divide the data into the training set and the testing set while learning dependence from data. We utilize the data from the testing set to measure the correctness of our model after it has been trained on the training set. According to empirical studies, the best results are obtained when 20-30% of the data is used for testing and the rest 70-80% is used for training.For this, while training our models with classifiers, we used 70% data for training and 30% data for testing, and we got good results.The lowest accuracy we found is around 98% and the highest around 99%.Some methods are used in the studies to increase the accuracy of the models. While increasing the accuracy of the models, the parameter values unique to each model were manipulated. The accuracy rates of the models were increased with the parameters "C" (arrangement parameter) in the Support Vector Machine model, the number of neighbors parameter in the k-Nearest Neighbor model, and the number of trees, the maximum depth of the tree, the number of samples for each partition, and the number of features in the Random Forest model.

4.5 User Interface Design

An example of a user interface that can be used for our project in the future is expected to be as seen in Figure 4.5. It is a very simple and single page interface with no menu. Under the activity column, there are the user's movements which are found in this project, and under the probability column, there is the probability rate of which activity was done. The activity with the highest rate becomes green and the activity output is written at the bottom. This user interface has been defined as a concept.

Figure 4.5 Android Phone User Interface Example

4.6 Software Development Methodology

We decided to use the agile method in this project. The agile method is a special approach to project management used in software development. This method helps teams respond to the unpredictability of software development processes. It uses incremental, iterative work sequences, commonly known as sprints. Sprint is the time allotted for a particular phase of a project. The project is considered complete when the sprints expire. General principles of Agile method; satisfy the customer, changing requirements should be adopted; delivery should be done as soon as possible, face-to-face communication, simplicity. We also used a software development framework which is extreme programming framework for agile. Extreme Programming aims to produce higher quality software, and higher quality of life for the development team. It is designed for teams of between two and 12 members, so it is ideally suited for student projects as our project are.

In the project we are developing, important tricks such as complex features, iterative plans, changing requirements, quick response, active feedback while creating the product, and teamwork should never be overlooked. Therefore, the agile methodology is the most suitable for our project. As shown in the below Figure 4.6, each stage we divide the part we are developing into small parts and plan, design, code and test. In this way, we can produce products that work continuously and receive feedback from the customer as well as respond to changing features.

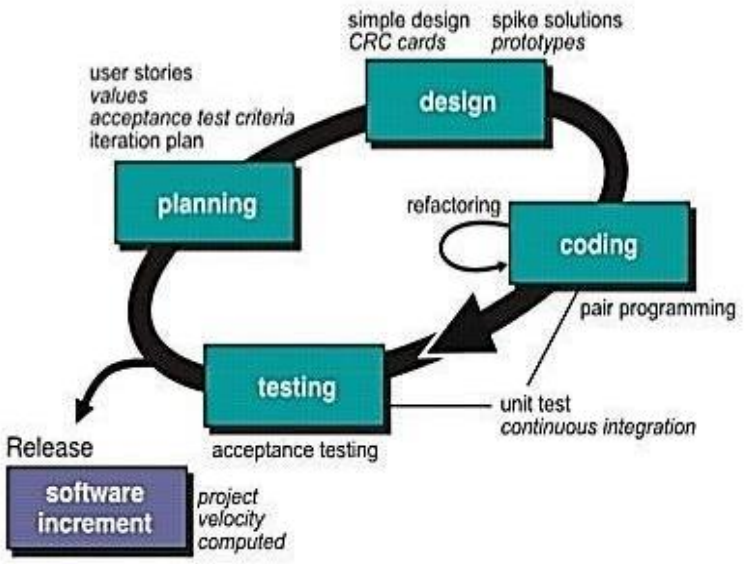


Figure 4.6 Extreme Programming Framework

4.7 Risks

As with any project, there are some risks. It is normal for the group to encounter the unexpected and taking precautions is essential. For this reason, while developing our project, we made some calculations and estimated the risks we might face. These can be listed as: Health ailments of the team members, exceeding the planned time period, failure in addressing priority conflicts, not keeping up with changing trends, waste of money and time when the right library and pattern is not selected in the project, errors in system architecture, training the system without correcting errors in datasets and unexpected departures within the team.

To find the function point of our project, we answered some questions in the domain characteristic table and complexity adjustment table, and these are shared in the Figure 4.7, 4.8. As a result, we found the function score of our project as 521.02.

MEASUREMENT PARAMETER	COUNT (value >= 0)	WEIGHTING FACTOR Simple	Average	Complex
Number of User Input	5	○	○	◉
Number of User Outputs	3	○	○	◉
Number of User Inquiries	30	○	◉	○
Number of Files	30	○	◉	○
Number of External Interfaces	1	○	◉	○

Figure 4.7 Domain Characteristic Table

ITEM	COMPLEXITY ADJUSTMENT QUESTIONS	SCALE No Influence 0	1	2	3	4	Essential 5
1	Does the system require reliable backup and recovery?	○	○	○	○	○	◉
2	Are data communications required?	○	○	○	○	○	◉
3	Are there distributed processing functions?	○	○	○	◉	○	○
4	Is performance critical?	○	○	○	○	○	◉
5	Will the system run in an existing, heavily utilized operational environment?	○	◉	○	○	○	○
6	Does the system require on-line data entry?	○	○	○	◉	○	○
7	Does the on-line data entry require the input transaction to be built over multiple screens or operations?	○	◉	○	○	○	○
8	Are the master files updated on-line?	○	◉	○	○	○	○
9	Are the inputs, outputs, files or inquiries complex?	○	○	○	○	◉	○
10	Is the internal processing complex?	○	○	○	○	○	◉
11	Is the code to be designed reusable?	○	○	○	○	◉	○
12	Are conversion and installation included in the design?	○	○	○	◉	○	○
13	Is the system designed for multiple installations in different organizations?	◉	○	○	○	○	○
14	Is the application designed to facilitate change and ease of use by the user?	○	○	○	○	◉	○

Figure 4.8 Complexity Adjustment Table

4.8 Timeline

In the figure shown below Figure 4.9, the timeline (Gantt chart) of the project is shared. According to this schedule, team members tried to complete the project development stages on time. While developing our project, we were able to adapt to this timeline and complete each stage on time. This enabled us to work in a planned and timely manner.

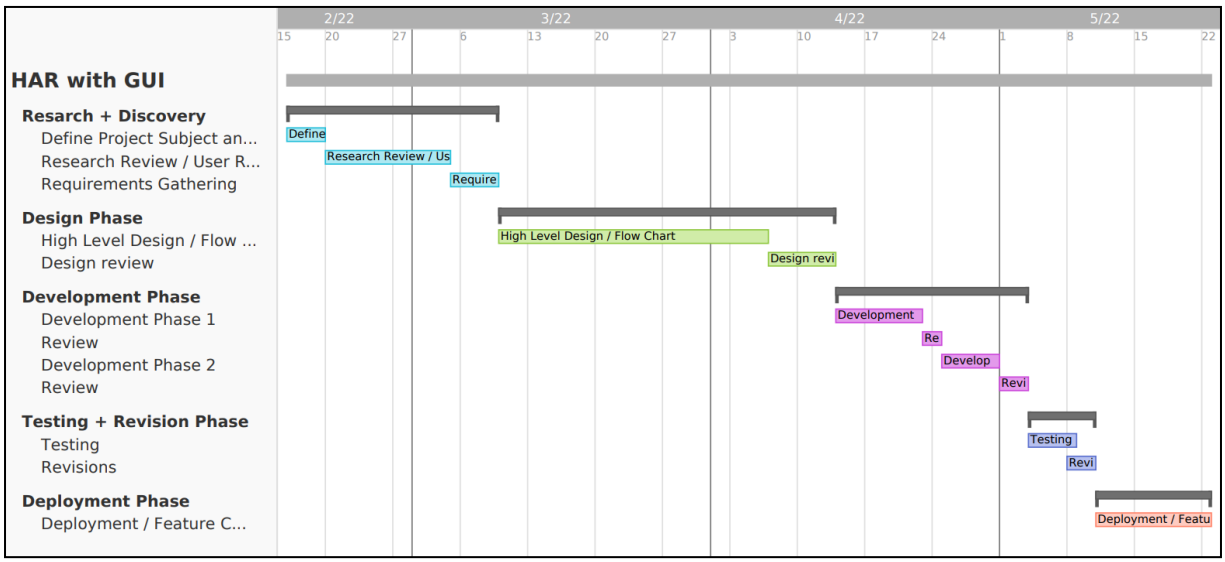


Figure 4.9 Gantt Chart

4.9 Budget

We used various programs and tools during the development of our project, some of which were free and had limited time to use. Besides, our dataset was ready; we didn't need to allocate a separate budget to collect the data. 4 computers were used during the development of this software and the average value of these computers is 700 dollars. The expenses incurred by all members of the group, except for the software, are calculated for 4 people and are shown in Figure 4.10 150 TL was spent for GitHub and similar sites used for the test phase of the program. With a monthly internet usage of 100 TL, the total cost of the group is 400 for 1 month and a total of 1600 TL has been incurred for 4 months of work. In the same way, electricity was calculated with an average of 400 TL and reflected in the table. The daily meal requirement, calculated from 40 TL, is 4800 per month, and the 4-month wage is reflected as 19200 TL. Finally, the monthly moving fee was calculated as 1200 TL for 4 people and the total budget was determined as 27000 TL.

1 ☆	Testing	Training Expense	150 TL
2 ☆	Internet	Workplace Expense	1600TL
3 ☆	Electric	Workplace Expense	5000TL
4 ☆	Transportation	Travel Expense	1200TL
5 ☆	Meal	Workplace Expense	19200TL

Figure 4.10 Budget Table

CHAPTER 5 IMPLEMENTATION & TEST

5.1 Implementation

This section of the document describes how the forecast classification algorithm project is implemented. This project is divided into two parts; model implementation and model accuracy improvement.

5.1.1 Model Implementation

Using the model application k-Nearest Neighbor, Support Vector Machines and Random Forest models, it is aimed to create an artificial intelligence that can classify 6 classes (walking, walking upstairs, walking downstairs, sitting, standing, laying). In this project, 3 models were selected to be used in the forecasting application. These models are trained on 2 basic platforms. These are our own computers using CPUs with Jupyter Notebook package in cloud (Colab) and Anaconda Navigator.

5.1.2 Import Modules

```
import pandas as pd

import numpy as np

import seaborn as sns

from sklearn.model_selection import train_test_split

from sklearn.model_selection import GridSearchCV

from sklearn.metrics import accuracy_score

from sklearn import metrics

from sklearn.ensemble import RandomForestClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.svm import SVC

import matplotlib.pyplot as plt
```

Figure 5.1 Libraries and Models Importing

5.1.3 Preparation of the Dataset

We load the dataset into our workspace with the "read_csv" function in the Pandas library as shown in Figure5.2. UCI HAR is used as data source. [10] In the code below, "random_state" is a fixed value for repeatability. Every time this model runs, "random_state" helps us to get the same values.

```
df = pd.read_csv("drive/My Drive/dataset.csv")

se = df.copy()

y = df["Activity"]

X = df.drop(["subject", "Activity"], axis = 1)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30, random_state = 94)
```

Figure 5.2 Preparing Dataset

The values of the data coming from the accelerometer and gyroscope are standardized between -1 and 1 as a result of the standardization process in the dataset we use, as shown in Figure 5.3.

	tBodyAcc-mean()-X	tBodyAcc-mean()-Y	tBodyAcc-mean()-Z	tBodyAcc-std()-X	tBodyAcc-std()-Y	tBodyAcc-std()-Z
0	0.288585	-0.020294	-0.132905	-0.995279	-0.983111	-0.913526
1	0.278419	-0.016411	-0.123520	-0.998245	-0.975300	-0.960322
2	0.279653	-0.019467	-0.113462	-0.995380	-0.967187	-0.978944
3	0.279174	-0.026201	-0.123283	-0.996091	-0.983403	-0.990675
4	0.276629	-0.016570	-0.115362	-0.998139	-0.980817	-0.990482

Figure 5.3 Example of Standardization First 5 elements of First 6 Columns

Two types of data are needed during the training of the model to get an accurate insight into the model. These are training data and test data. Our dataset provided us the UCI HAR dataset in one part. But the dataset had to be divided into 2 parts. Thus, it was necessary to separate the data set into training and test parts. From the scikit-learn library, we have defined the ratio of the dataset to be included in the test split as 0.3 with the "train_test_split" function. And we divided the dataset into two different sets as test and training. As a result, the test set was divided into 2 different datasets as 30% and the training set as 70%, as shown in Figure 5.4.

```
print(X_train.shape, X_test.shape)
```

```
(7209, 561) (3090, 561)
```

Figure 5.4 Train and Test Data Size

5.2 Training Algorithms

The accuracy rate was calculated on the data set with 3 different models. These models are SVM, kNN and RF respectively. The accuracy of the models has been increased by adjusting the hyper parameters of all models.

5.2.1 Support Vector Machine

As shown in Figure 5.5, "SVC" was imported from the sci-kit learn library to use the SVM model. Then, the training and test sets to be used in the model are divided into data sets with the "train_test_split" function. These clusters are separated by a ratio of 70-30. The value "random_state" is defined arbitrarily 70 in this model. Because we need the same random order in our work [5].

```
from sklearn.svm import SVC
```

```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.30, random_state = 70)
```

Figure 5.5 Splitting Data

Another process is to set up the model. We did this with the "fit" function. We fit the training parts of the x and y sets. We defined this model setup with the name "svm_model". Then we make the prediction using the set "X_test" with the "predict" function and save it with the name "y_pred". Finally, we find the prediction-accuracy ratio of this model by using the "y_test" and "y_pred" clusters in the accuracy score metric. The setup of the model was found to be 96.40% regardless of the hyper parameter settings, as shown in Figure 5.6.

```
from sklearn.metrics import accuracy_score

svm_model = SVC().fit(X_train, y_train)

y_pred = svm_model.predict(X_test)

accuracy_score(y_test, y_pred)

0.9640776699029127
```

Figure 5.6 SVM Model Setup, Estimation and Accuracy Rate

5.2.2 K-Nearest Neighbor

To utilize the kNN model, first import "KNeighborsClassifier" from the sci-kit learn package. The "train test split" function is then used to split the training and test sets into data sets for the model. The distance between these clusters is 70-30. In this model, the value "random state" is set to 8 at random, as shown in Figure 5.7. Because our work requires the same random arrangement [4].

```
from sklearn.neighbors import KNeighborsClassifier

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.30, random_state = 8)
```

Figure 5.7 Splitting Data

Another step is to create the model. We used the "fit" tool to accomplish this. The x and y sets' training pieces were fitted. The name "kNN model" was used to identify this model setup. Then, using the set "X test" and the "predict" function, we create the prediction and save it as

"y pred." Finally, we use the "y test" and "y pred" clusters in the accuracy score metric to calculate the model's prediction-accuracy ratio. Regardless of the hyper parameter parameters, the model setup was found to be 95.55 percent correct, as shown in Figure 5.8.

```
from sklearn.metrics import accuracy_score

knn_model =knn.fit(X_train, y_train)

y_pred = knn_model.predict(X_test)

accuracy_score(y_test, y_pred)

0.955663430420712
```

Figure 5.8 kNN Model Setup, Estimation and Accuracy Rate

5.2.3 Random Forest

Import "RandomForestClassifier" from the sci-kit learn package to use the RF model. The model's training and test sets are then split into data sets using the "train test split" function. These clusters are 70-30 distance apart. The value "random state" is set to 94 at random in this model, as shown in Figure 5.9. Because our job necessitates the same haphazard setup.

```
from sklearn.ensemble import RandomForestClassifier

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30, random_state = 94)
```

Figure 5.9 Splitting Data

The model must also be created. This was accomplished using the "fit" tool. The training components for the x and y sets were installed. This model arrangement was given the moniker "rf_model." The prediction is then created and saved as "y pred" using the set "X test" and the "predict" function. Finally, we construct the model's prediction-accuracy ratio using the "y test" and "y pred" clusters in the accuracy score metric. The set-up of the model was 96.99% before the hyper parameter tuning, as you can see in Figure 5.10.

```
from sklearn.metrics import accuracy_score

rf_model = RandomForestClassifier().fit(X_train, y_train)

y_pred = rf_model.predict(X_test)

accuracy_score(y_test, y_pred)

0.96957928802589
```

Figure 5.10 Random Forest Model Setup, Estimation and Accuracy Rate

5.3 Plotting Accuracy Score and Training Test Sizes

In the graph below, the accuracy score of the train set is marked as “Training Score” and the accuracy score of the test set as “Cross-Verification Score”. Up to a training size of about 0.7, the model's training score is much higher than the test score. Therefore, if our current dataset has a training rate of much less than 0.7, increasing the rate of more training will increase generalization. However, after level 0.7, increasing the training size of the model will not be of much benefit. Keeping the training size of the model at the optimum level is a priority. Teaching the training size too much can overfit the model. As seen in the Figure 5.11, it is observed that learning slows down in the 0.7 band and enters the saturation process. As a result, we chose this test size [9].

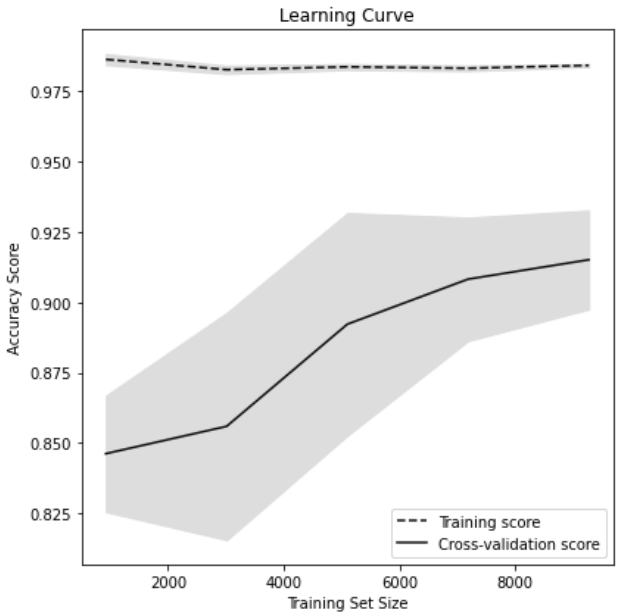


Figure 5.11 Accuracy Score and Training Test Size Graph

5.4 Generating Confusion Matrix

Confusion matrix is a table used to describe the performance of a classification model on a set of test data for which the actual values are known. In this section, confusion matrix was drawn for SVM, kNN and Random Forest models, respectively.

To draw a confusion matrix, the required module is imported from the "sklearn.metrics" library, as shown in Figure 5.12.

```
from sklearn.metrics import confusion_matrix
```

Figure 5.12 Import Confusion Matrix

5.4.1 Confusion Matrix Tree for Models

To create a confusion matrix tree, dataset "y_test" is placed on the x-axis and dataset "y_pred" is placed on the y-axis, as shown in Figure 5.13.

```
cm_tree = confusion_matrix(y_test.to_numpy(), y_pred)
cm_tree
```

Figure 5.13 Confusion Matrix Code

They are listed on the X-axis as laying, sitting, standing, walking, walking downstairs and walking upstairs, respectively. On the Y axis, the classification examples were listed as the opposite. First of all, prediction classes must be passed through test classes. Second, "y_test" is passed to the "cm_tree" object in the form of a list, using the numpy library. This function returns the index with the largest value along the axes of a tensor as shown in Figure 5.14, 5.15, 5.16. After that, the result of the predictions and the return value are given to the "configuration_matrix" function. We recorded this action in the "cm_tree" session object. Session objects allocate system resources to perform mathematical operations. Finally, printing is used to print the confusion matrix.

```
array([[575,   0,   0,   0,   0,   0],
       [  0, 504,  53,   0,   0,   0],
       [  0,  43, 515,   0,   0,   0],
       [  0,   0,   0, 517,   0,   2],
       [  0,   0,   0,   2, 397,   5],
       [  0,   0,   0,   3,   3, 471]])
```

Figure 5.14 CM Tree for SVM

```
array([[573,   0,   0,   0,   0,   0],
       [  2, 485,  61,   0,   0,   2],
       [  0,  53, 529,   0,   0,   0],
       [  0,   0,   0, 504,   1,   1],
       [  0,   0,   0,   9, 395,   3],
       [  0,   0,   0,   4,   1, 467]])
```

Figure 5.15 CM Tree for kNN

```
array([[602,   0,   0,   0,   0,   0],
       [  0, 479,  34,   0,   0,   0],
       [  0,  18, 537,   0,   0,   0],
       [  0,   0,   0, 500,   4,   3],
       [  0,   0,   0,   5, 412,   9],
       [  0,   0,   0,   1,  10, 476]])
```

Figure 5.16 CM Tree for RF

5.4.2 Visualizing Confusion Matrix

We used the seaborn library to show the confusion matrix in a more understandable way. The code box below contains all the code needed to create a visualized confusion matrix. In the first line, the classification list thrown into the "sorted" function is arranged in order to position the y-axis as the opposite of the x-axis in the confusion matrix. In the second line, the labels of the "df_tree" environment object are assigned. On the other line, Matplotlib's shape function is called and the figsize argument specifies the width and height of the shape in inches. Heatmap function of Seaborn library plots rectangular data as a color encoded matrix. The first argument given in the function is the confusion matrix, the second argument specifies that the

data values should be written in each cell, the third argument specifies the value mappings, and the “fmt” argument is for printing the values in each cell normally and not in scientific notation. The last argument specifies which color should be used to draw the matrix. "plt.x_label" indicates the name of the x-axis, "plt.y_label" indicates the name of the y-axis, and "plt.title" indicates the name of the chart. The "show" function is used to show the graph in the last line as shown in Figure 5.17.

```
categories = sorted(df["Activity"].unique(), reverse=False)
df_tree = pd.DataFrame(cm_tree, index = categories, columns = categories)

plt.figure(figsize = (10,6))
sns.heatmap(df_tree, annot=True, annot_kws={"size": 12}, fmt="g", cmap="YlGnBu")
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.title("Confusion Matrix Classification Trees for SVM Model")
plt.show()
```

Figure 5.17 Plot functions for Drawing Confusion Matrix

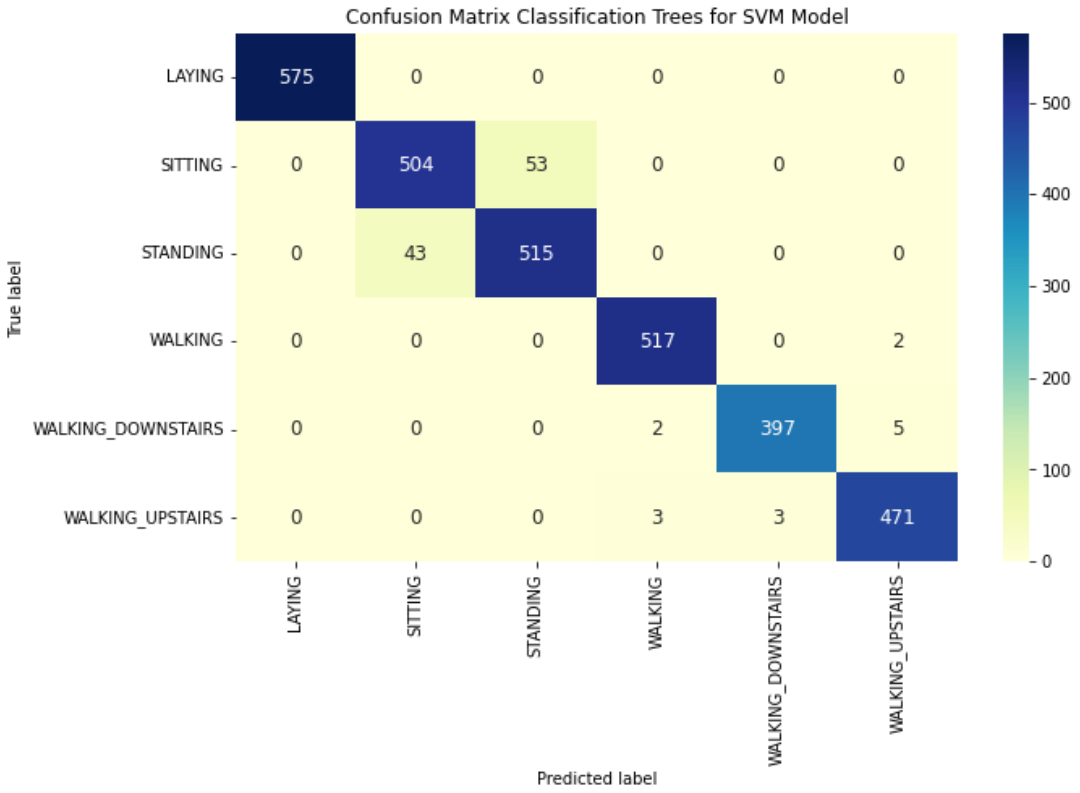


Figure 5.18 Confusion Matrix Classification Trees for SVM Model

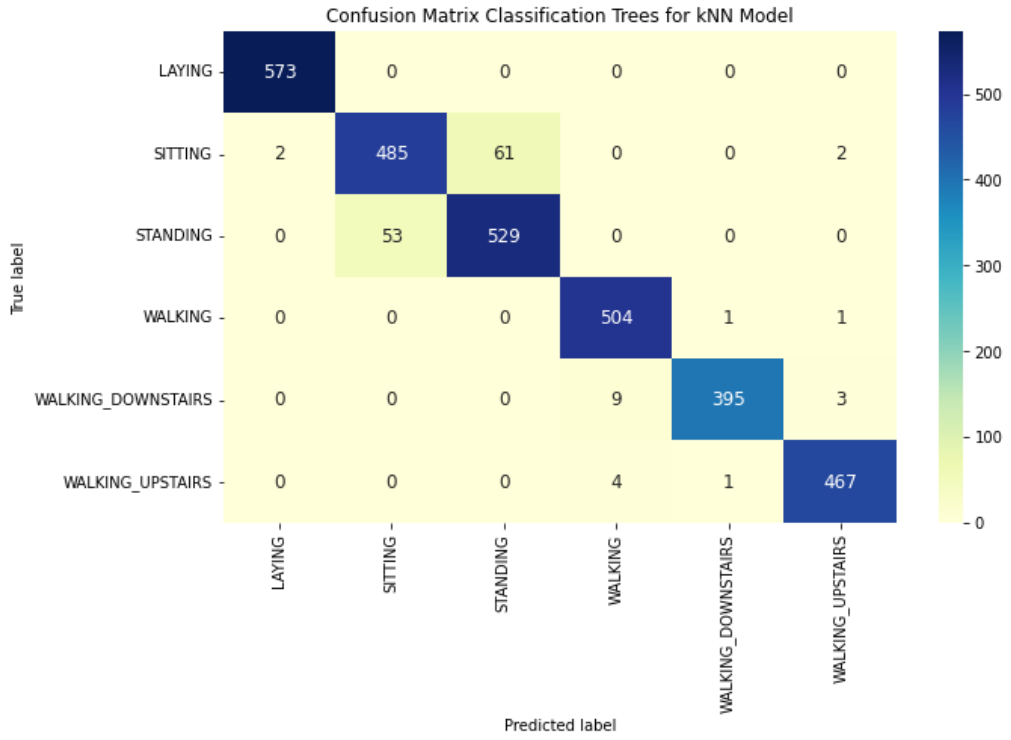


Figure 5.19 Confusion Matrix Classification Trees for kNN Model

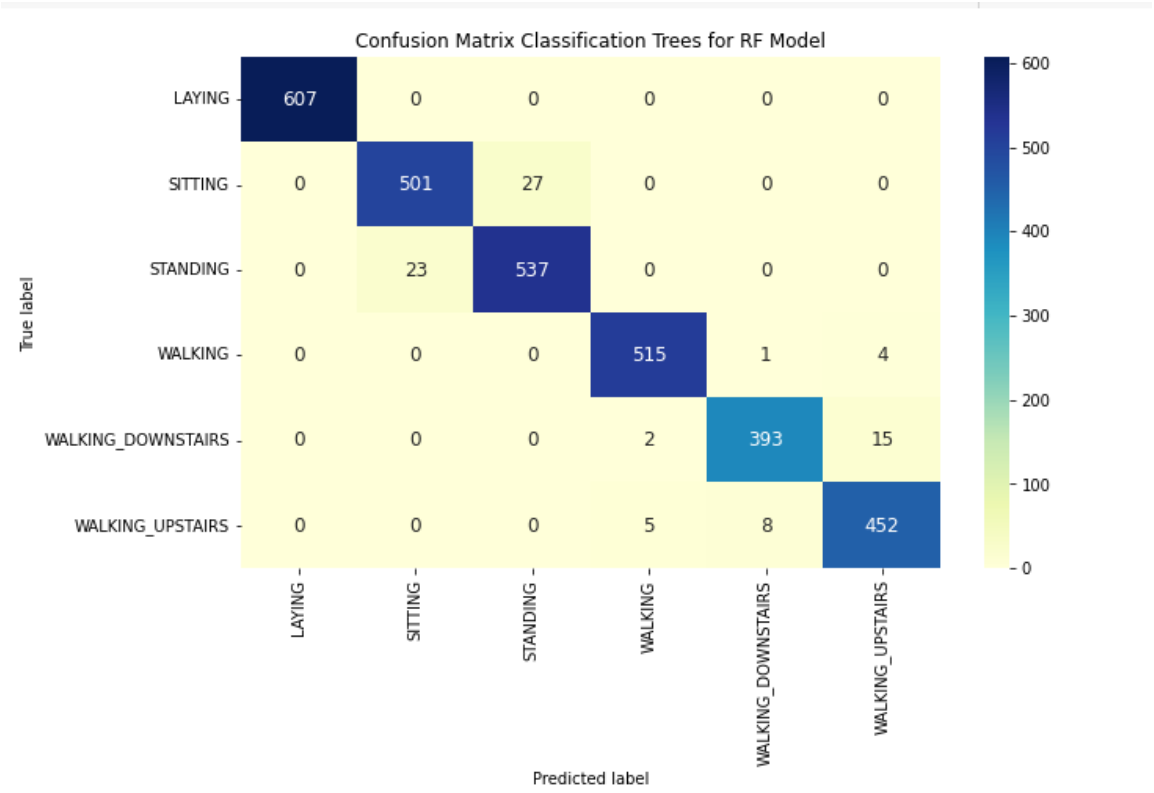


Figure 5.20 Confusion Matrix Classification Trees for RF Model

5.5 Test

In this project, every hyper parameter optimization is a test. Different hyper parameters are tested with different functions in different scenarios to get a better classifier model. During the development of the models, nonlinear classification methods were studied. Thus, each model has its own hyper parameters. In addition, there are parameters that must be adjusted very precisely to achieve the best results despite the large dataset.

5.6 Hyper Parameter Optimization with Gridsearch CV

A separate model is built with all combinations for the hyper parameters and their values desired to be tested in the model, and by iteratively changing the hyper parameters one after the other, the performance of the model is observed and the most suitable hyper parameter group for the model has been tried to be selected.

```
from sklearn.model_selection import GridSearchCV
```

Figure 5.21 Importing GridSearchCV Library

5.6.1 SVM for Hyper Parameter Tuning

5.6.1.1 First Iteration

First, the hyper parameters of the kernel parameter are; linear, rbf, sigmoid and poly. Graphs were drawn to observe the effects of the C parameter placed on the x-axis in the model, with the kernel hyper parameters on the y-axis. Looking at the Figures 5.22-25 below, graphs with kernel parameters "linear" and "poly" have the highest accuracy rates. In this part, the kernel parameter and the C parameter are combined. In addition, the C value was observed in the "poly" and "linear" parameters with the highest values. As a result of the observations, it was determined that the C value showed the observation of the data after 25 in both graphs.

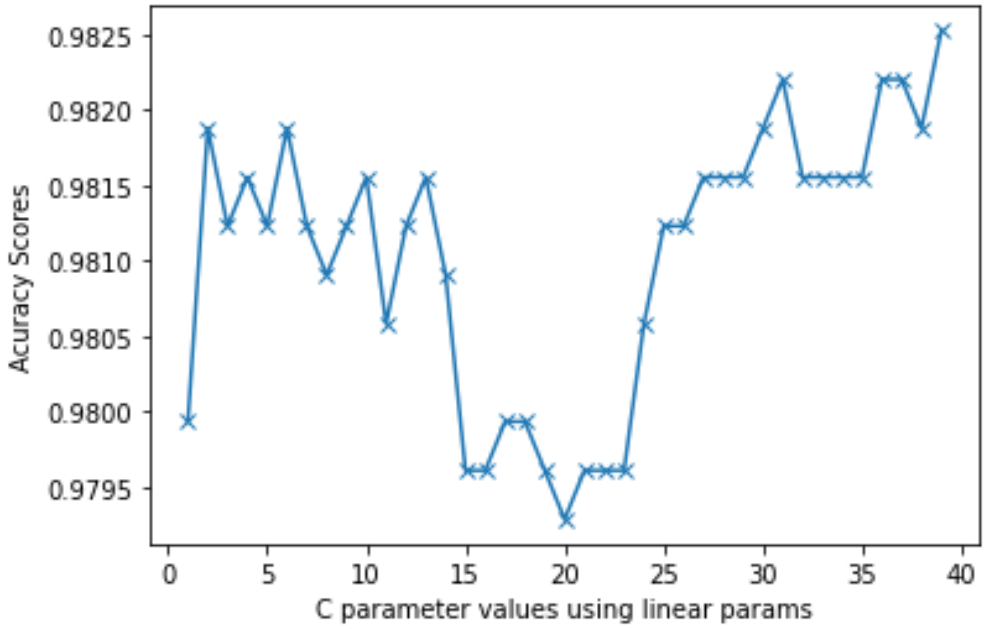


Figure 5.22 Accuracy Graph with C Parameter using Linear Params

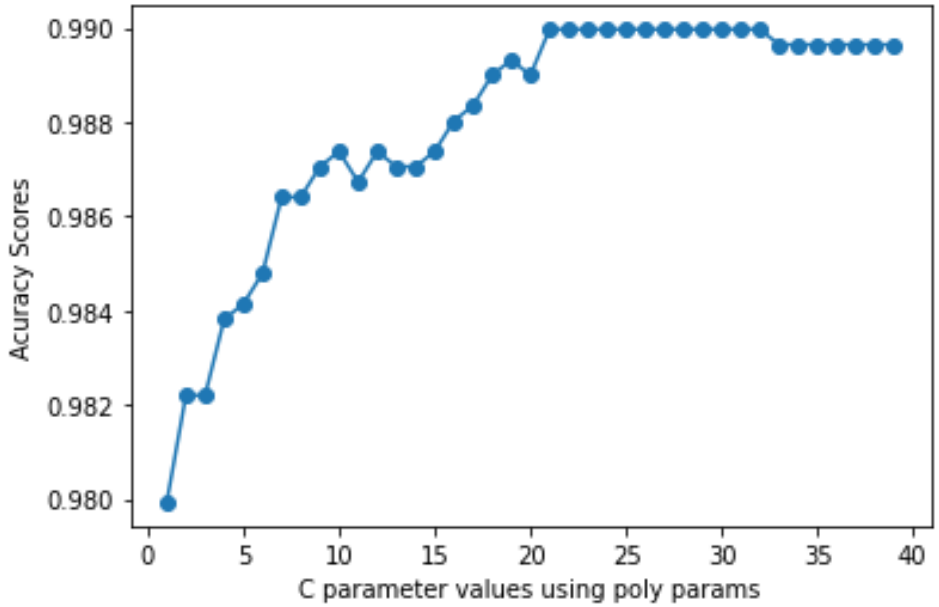


Figure 5.23 Accuracy Graph with C Parameter using Poly Params

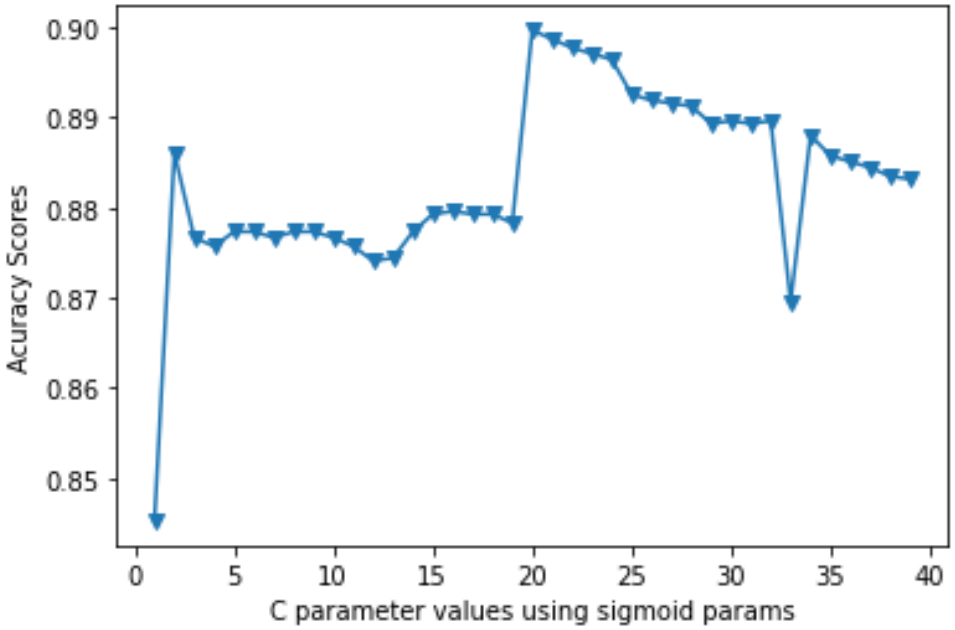


Figure 5.24 Accuracy Graph with C Parameter using Sigmoid Params

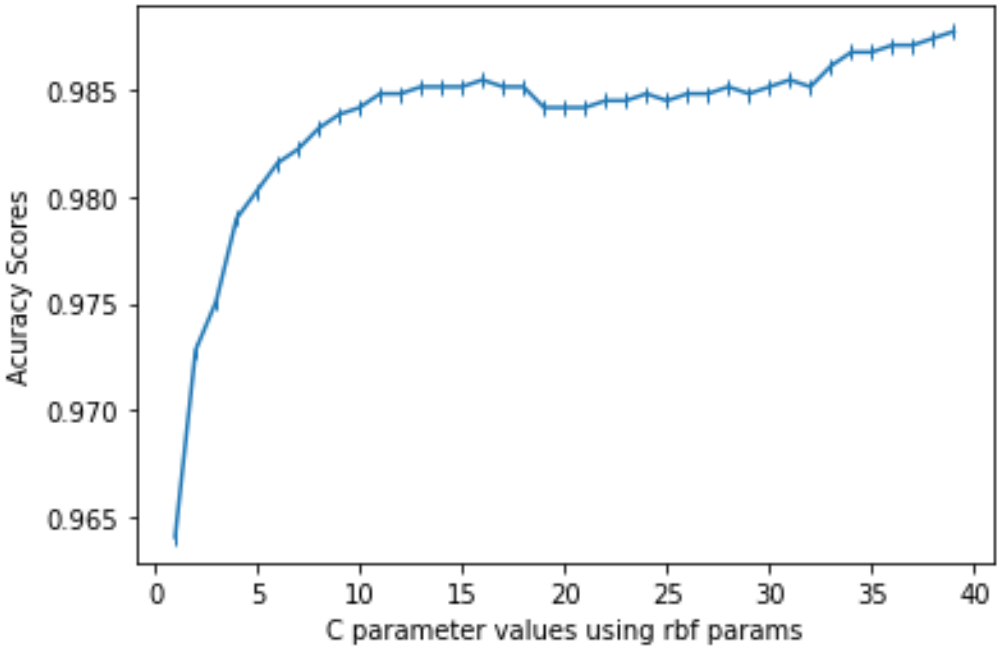


Figure 5.25 Accuracy Graph with C Parameter using rbf Params

5.6.1.2 Second Iteration

In the next step, reduction was made in kernel and C parameters. Work continued with linear and poly in the kernel parameter. In addition, as a result of the tests, C values for the other iteration were observed between 25-50. As a result, C values were determined to be 30, 35, 40, 45 and 50 in the other iteration as shown in Figure 5.26, 5.27.

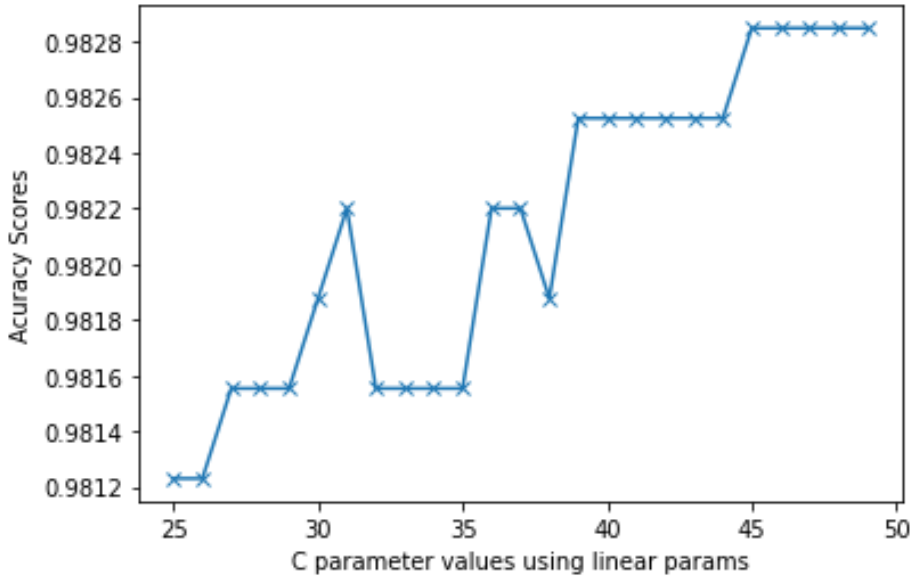


Figure 5.26 Accuracy Graph with C Parameter using Linear Params

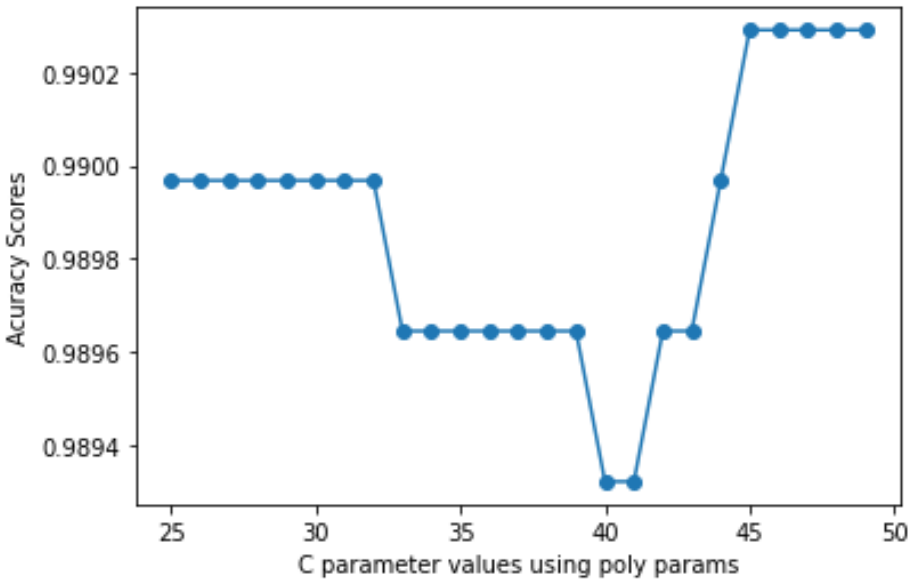


Figure 5.27 Accuracy Graph with C Parameter using Poly Params

5.6.1.3 Third Iteration

In the other iteration, models were built with the GridSearch CV method with all parameter values. This time, the model is combined with 3 parameters. The hyper parameter values of the parameters are given below in the "svc_params" environment object. The model is set to the "svc_cv_model" object using the GridSearch method. The first parameter is the environment object where we save the model, the second parameter contains hyper parameter values, the third parameter expresses how many floors the model will be, the last parameter gives information while the model is being built. In the last line, the model is built as shown in Figure 5.28.

```
svc_params = {"C" : [30,35,40,45,50],
              "kernel" : ["poly", "linear"],
              "degree" : [1,2,3,4,5,6,7,8,9,10]}
```

```
svc_cv_model = GridSearchCV(svm_model, svc_params, cv = 10, verbose = 2)
```

```
svc_cv_model.fit(X_train, y_train)
```

Figure 5.28 Build a Model with Hyper Parameters

The 10-fold model installation process was carried out with 100 candidates. After 1000 tests in total, all models were installed separately with GridSearch CV as shown in Figure 5.29.

```
Fitting 10 folds for each of 100 candidates, totalling 1000 fits
[CV] END .....................C=30, degree=1, kernel=poly; total time=   0.9s
[CV] END .....................C=30, degree=1, kernel=poly; total time=   0.9s
[CV] END .....................C=30, degree=1, kernel=poly; total time=   0.9s
```

Figure 5.29 Fitting Candidates

The parameters that give the best results among 100 different models are listed in Figure 5.30.

```
svc_cv_model.best_params_
```

```
{'C': 30, 'degree': 6, 'kernel': 'poly'}
```

Figure 5.30 Best Params for SVM

The model was rebuilt and trained with Best hyper parameters. These values were found 30 for the "C" parameter, 6 for the "degree" parameter and poly for the "kernel" parameter. As a result of training and prediction, the result of hyper parameters giving the highest accuracy score for SVM is 99.15%, as shown in Figure 5.31.

```
svc_tuned = SVC(degree = 6, kernel = "poly", C = 30).fit(X_train, y_train)
```

```
y_pred = svc_tuned.predict(X_test)
```

```
accuracy_score(y_test, y_pred)
```

```
0.9915857605177993
```

Figure 5.31 Fitting Best Params with SVM

5.6.2 kNN For Hyper Parameter Tuning

The parameters to be tuned in the kNN model are as follows; "algorithm", "n_neighbors", "metric", "weights".

5.6.2.1 First Iteration

First of all, the hyper parameter values of the "metric" parameter are; "manhattan", "minkowski" and "chebyshev". In the model, the effect of the hyper parameter values of the metric parameter was observed, with the other parameter values remaining as default. As a result, the "manhattan" hyper parameter showed better accuracy than the others as shown in Figure 5.32.

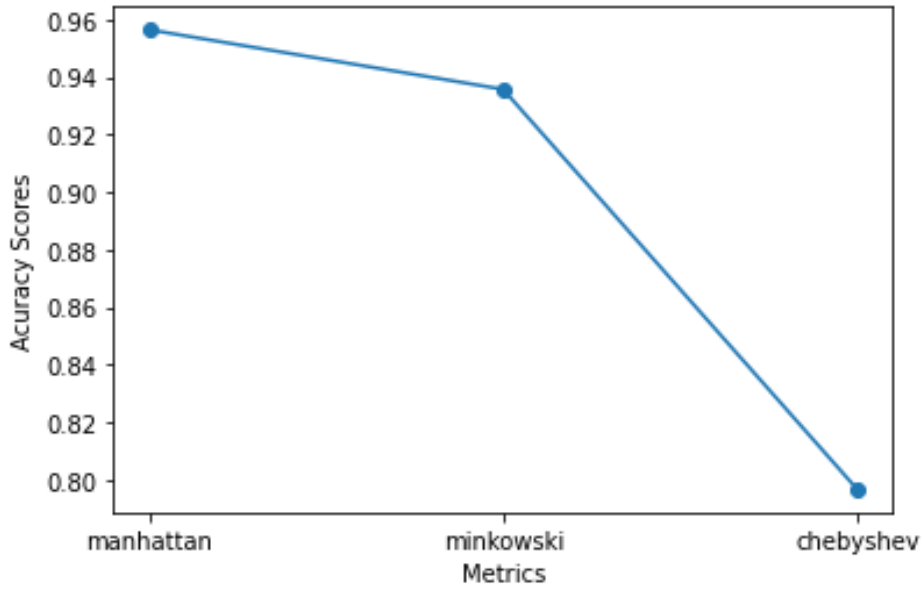


Figure 5.32 KNN Metric Params Accuracy Scores

5.6.2.2 Second Iteration

In the second iteration, the "metric" parameter is fixed with "manhattan". Here, it is desired to observe the results of the number of neighbors. However, the number of neighbors is shown in Figure 5.33 from 1 to 40. As a result, in the next iteration, the number of neighbors will be observed between 1 and 5.

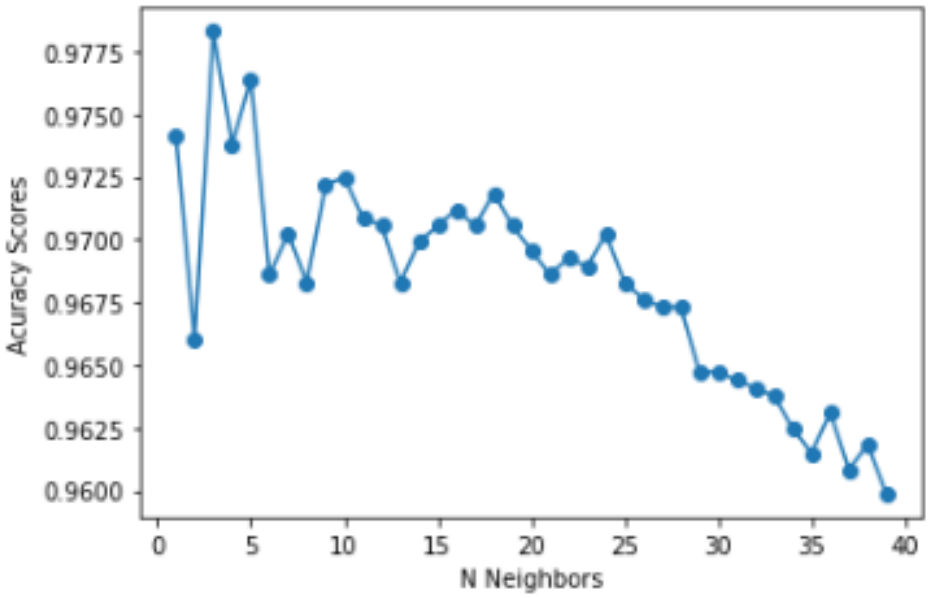


Figure 5.33 N Neighbors with Accuracy Score

5.6.2.3 Third Iteration

In the last iteration, the hyper parameter values of the "weights" parameter were studied in addition to the specified parameters. These hyper parameters were changed from "uniform" to "distance" and observed. By looking at Figures 5.34, 5.35, the ranges of the best parameters for kNN were selected.

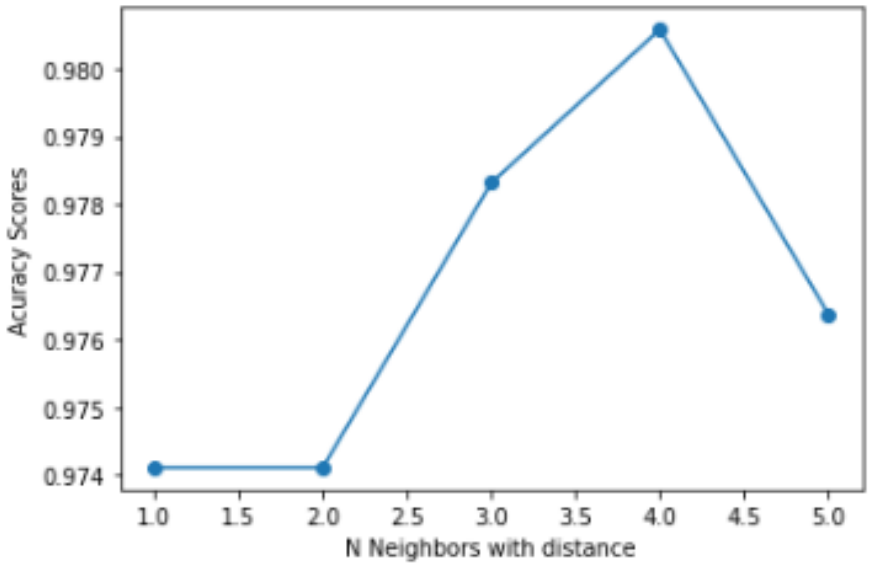


Figure 5.34 N Neighbors with Distance

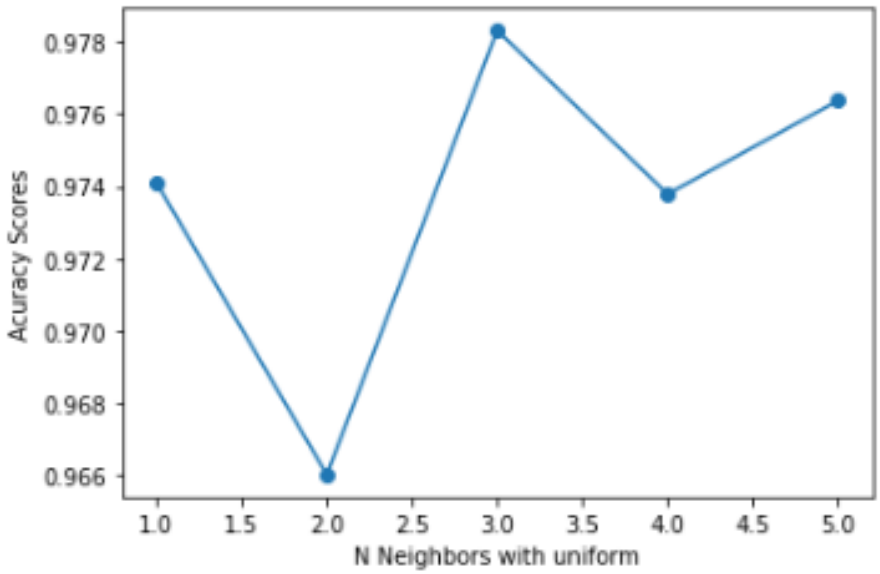


Figure 5.35 N Neighbors with Uniform

Finally, considering all hyper parameter values, a 10-fold model was established with the GridSearch CV method as shown in Figure 5.36. In total, 100 tests were performed in the last step as shown in Figure 5.37. The model was rebuilt and trained with Best hyper parameter values. As a result, the best hyper parameter values for kNN are 4 for the number of neighbors, distance for the weight, and manhattan for the metric. With these hyper parameters, our accuracy rate was found to be 98.05%, as shown in Figure 5.39.

```
knn_params = {"n_neighbors" : [1,2,3,4,5],
              "weights" : ["distance", "uniform"],
              "metric" : ["manhattan"]}

knn_cv_model = GridSearchCV(knn, knn_params, cv = 10, verbose = 2)

knn_cv_model.fit(X_train, y_train)
```

Figure 5.36 Build a Model with Hyper Parameters

```
knn_cv_model.fit(X_train, y_train)
[CV] END ..metric=manhattan, n_neighbors=3, weights=distance; total time=   1.5s
[CV] END ..metric=manhattan, n_neighbors=3, weights=distance; total time=   1.5s
[CV] END ..metric=manhattan, n_neighbors=3, weights=distance; total time=   1.7s
```

Figure 5.37 Fitting Candidates

```
knn_cv_model.best_params_

{'metric': 'manhattan', 'n_neighbors': 4, 'weights': 'distance'}
```

Figure 5.38 Best Params for kNN

```
knn_tuned.fit(X_train, y_train)
```

```
KNeighborsClassifier(metric='manhattan', n_neighbors=4, weights='distance')
```

```
y_pred = knn_tuned.predict(X_test)
```

```
accuracy_score(y_test, y_pred)
```

```
0.9805825242718447
```

Figure 5.39 Fitting best Params with kNN

5.7 Increasing the Accuracy of the Random Forest Classifier

To increase the accuracy of the Random Forest Classifier model we used Recursive feature elimination and features importance.

5.7.1 Recursive Feature Elimination

Recursive feature elimination is a type of backward feature elimination in which we first fit our model using all of the features in a given set, then remove the least significant features one by one, and keep re-fitting with different parameters, until we have the desired number of features, which is set by the parameter "n_features_to_select", which we can import from Sklearn,feature_selection library as you can see in Figure 5.40.

```
from sklearn.feature_selection import RFE, SelectFromModel
```

Figure 5.40 Importing RFE Library

To properly use the Recursive feature elimination, we created a new model with the random forest classifier.The RFE library was used to initialize a tree variable, and we chose the following parameters: the estimator parameter, which represents the new model, and the "n features to select"parameter, which represents the number of features to be selected. We fitted the tree with the previous 70% training sets "X train, y train", then initialized a "ypred" variable with the machine prediction of the activities based on the set on the 30% testing data "X test", and compared it to the "ytest" to determine the accuracy of the machine prediction. As shown in Figure 5.41.

```
: model_tree4=RandomForestClassifier(n_estimators=100,random_state=42)

: tree4=RFE(estimator=model_tree4,n_features_to_select=60,step=1)

: X_train_tree4=tree4.fit_transform(X_train,y_train)

: y_pred4=tree4.predict(X_test)

: metrics.accuracy_score(y_test,y_pred4)

: 0.9750809061488673
```

Figure 5.41 Training RFE Model with 60 Selected Features

To observe how the Recursive feature elimination works, we used the ".get support" function to print the selected features that are represented as TRUE, As shown in Figure 5.42. It took about two hours to try one parameter because it goes through all the features and deletes them one by one, because of that we used important features which is faster than Recursive feature elimination.

```
print(tree4.get_support())

[False False False  True False False False False False  True False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False  True False False  True  True  True False False False False False
 False  True  True  True  True  True  True False  True  True False False
 False False False  True False  True  True False False  True  True  True
 False  True  True  True  True False False False False False False False
 False False  True False False False False False False False False  True
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False  True False False  True False False
 False False False False False False False  True False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False  True False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False  True  True False
 False False False False False  True False False False False  True  True
 False False False False False False False False False False  True False
  True False False False  True  True False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False  True False False  True False False False False
 False False False False False  True False False False False False False
 False False False False False False False False False False False False
 False False  True False False False False False False False  True False
 False False  True False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False  True False False
 False False False False False  True False False False  True False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False  True False  True  True False False False False
 False False False False False False False False False False False False
  True False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False  True  True
  True False False  True  True False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False  True  True  True]
```

Figure 5.42 Best 60 Selected Features Represented as TRUE

5.7.2 Feature Importance

Feature importance refers to techniques that compute a score for each of the input features for a given model. These scores simply represent the importance of each feature. A higher score indicates that the specific feature will have a greater impact on the model used to predict a specific variable. The ".features_importances_" function gives an array of the scores. As we can see in the Figure 5.43 the highest score of a specific feature was 4.1943.

```
: model.feature_importances_

: array([4.19437333e-04, 2.38606285e-04, 1.90308854e-04, 4.34724997e-03,
       6.55174212e-04, 5.29024393e-04, 7.04725226e-03, 5.61904624e-04,
       4.70258300e-04, 1.08856024e-02, 2.25109000e-04, 3.57121632e-04,
       3.31734262e-04, 2.80241667e-04, 1.63989534e-04, 4.70157124e-04,
       2.43070232e-03, 7.93425305e-04, 3.44131675e-04, 3.09279508e-03,
       3.71730840e-04, 1.88426823e-04, 1.42636382e-03, 2.86750294e-04,
       4.46537929e-04, 3.56138685e-04, 2.47041647e-04, 2.33680242e-04,
       3.39846572e-04, 1.93003768e-04, 1.61481001e-04, 2.79801768e-04,
       1.75674539e-04, 1.65017951e-04, 1.85560524e-04, 1.66941134e-04,
       3.36772090e-04, 5.28074998e-03, 1.64242783e-03, 1.35655818e-03,
       2.17951769e-02, 2.02385734e-02, 8.10287535e-03, 2.64214222e-03,
       2.43331323e-03, 7.65342181e-04, 1.28327943e-03, 1.52895277e-03,
       8.42478565e-04, 2.77856348e-02, 2.65899001e-02, 1.03657644e-02,
       3.53364005e-02, 1.95926362e-02, 7.54702705e-03, 3.02279751e-03,
       2.15600757e-02, 1.46915751e-02, 7.89237708e-03, 1.12797106e-03,
       4.63683891e-04, 6.21506986e-04, 1.45545446e-03, 6.06478986e-03,
       2.21118262e-04, 5.62347128e-03, 6.76939560e-03, 9.34229801e-03,
       1.32255862e-03, 7.00926137e-03, 9.63893821e-03, 7.31783364e-03,
       5.86883745e-03, 9.05748135e-03, 8.49937564e-03, 7.86633056e-03,
       4.52070897e-03, 2.78693947e-03, 8.32577423e-04, 1.20715602e-03,
       1.19738695e-04, 1.02528515e-04, 9.99750093e-05, 6.73422763e-03,
       5.01561109e-04, 5.72694563e-04, 8.19689242e-03, 5.37437322e-04,
       5.80322822e-04, 4.12363901e-03, 4.19758475e-04, 6.94148397e-04,
       3.15741393e-04, 3.25276819e-04, 1.78131326e-04, 8.38431812e-03,
       8.78487910e-03, 6.21750268e-03, 3.92547267e-04, 8.27205129e-03,
       8.36296992e-03, 3.54962149e-04, 1.09199872e-03, 3.32497495e-04,
       6.84039333e-04, 3.60279134e-04, 1.62877857e-04, 1.57354691e-04,
```

Figure 5.43 Features Importance Scores Array

From the Pandas library we used "pd.series" to map the column names "X.columns" to the random forest feature importance "model.feature_importances_" and assign it into"feature_imp" variable, the aim of this mapping is to be able to see which column has the highest score, As you can see in Figure 5.44.

```
feature_imp = pd.Series(model.feature_importances_, index=X.columns).sort_values(ascending=False)
```

Figure 5.44 Mapping Column Names to the RF Features Importance

From the Matplotlib Library we used the Subplots function to create a figure, we set a title for the figure using the "set_title"function, we set x and y labels using "set_xlabel,set_ylabel" functions, and we used the Barplot function to set the x and y parameters of the figure from the Seaborn library to observe the most important features and the least according to thescores, As shown in Figure 5.45. To see the whole Figure 5.46, 5.47 check [10].

```
f, ax = plt.subplots(figsize=(15,110))
ax = sns.barplot(x=feature_imp, y=feature_imp.index)
ax.set_title("Visualize feature scores of the features")
ax.set_yticklabels(feature_imp.index)
ax.set_xlabel("Feature importance score")
ax.set_ylabel("Features")
plt.show()
```

Figure 5.45 Creating Features Importance Code

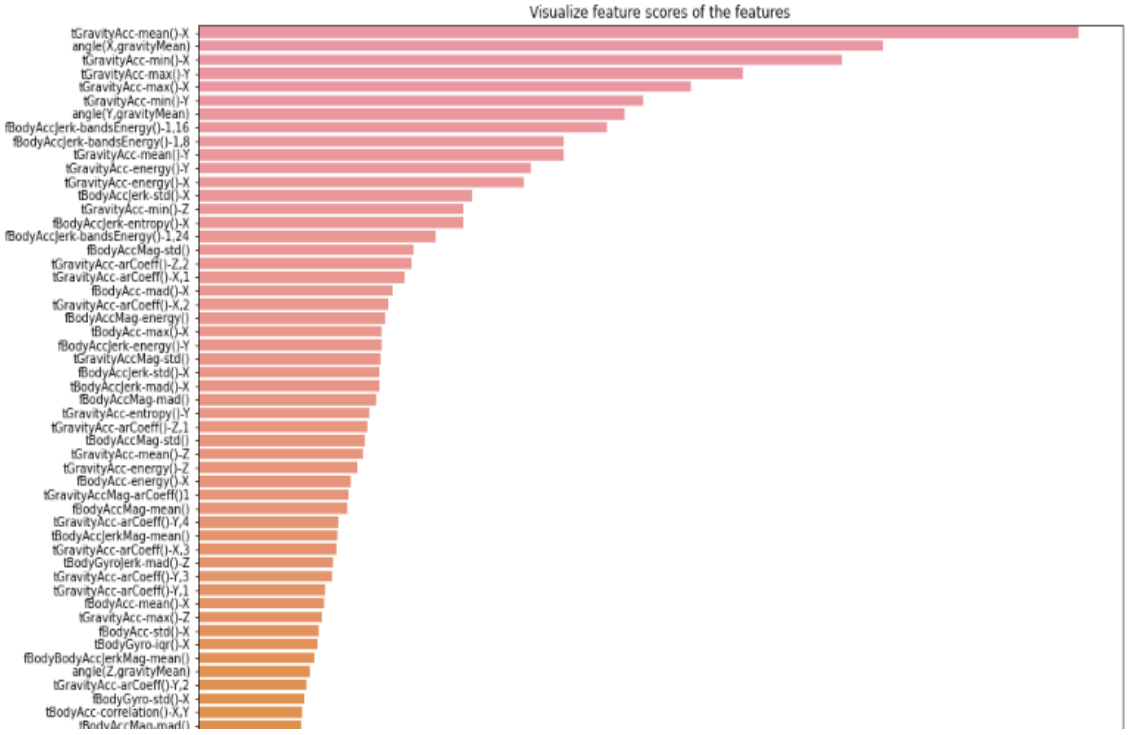


Figure 5.46 the Head of the Features Importance Figure

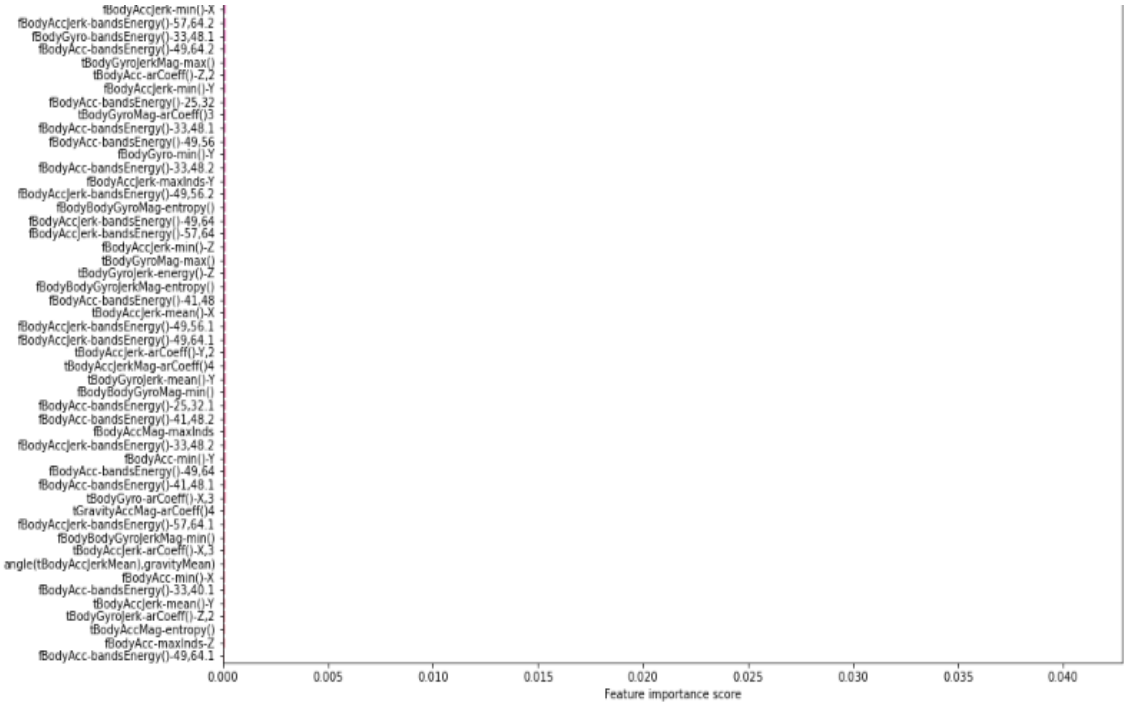


Figure 5.47 the Tail of the Features Importance Figure

From Figure 5.47 we found out that more than half of the columns have low scores, we have tried many parameter intervals as you can see in these Figures 5.48-50, and fitted them but the accuracy was not increasing.

```
feature=feature_imp.index[0:100]
```

```
B=data[feature]
A=data['Activity']
```

```
B_train, B_test , A_train , A_test =train_test_split(B,A,random_state=42)
```

```
model=RandomForestClassifier()
```

```
model.fit(B_train,A_train)
```

```
RandomForestClassifier()
```

```
pred=model.predict(B_test)
```

```
model.score(B_test,A_test)*100
```

```
97.7864077669903
```

Figure 5.48 Training the First 100 Important Features

```
feature=feature_imp.index[0:200]
```

```
B=data[feature]
A=data['Activity']
```

```
B_train, B_test , A_train , A_test =train_test_split(B,A,random_state=42)
```

```
model=RandomForestClassifier()
```

```
model.fit(B_train,A_train)
```

```
RandomForestClassifier()
```

```
pred=model.predict(B_test)
```

```
model.score(B_test,A_test)*100
```

```
97.7864077669903
```

Figure 5.49 Training the First 200 Important Features

```
feature=feature_imp.index[0:250]
```

```
B=data[feature]
A=data['Activity']
```

```
B_train, B_test , A_train , A_test =train_test_split(B,A,random_state=42)
```

```
model=RandomForestClassifier()
```

```
model.fit(B_train,A_train)
```

```
RandomForestClassifier()
```

```
pred=model.predict(B_test)
```

```
model.score(B_test,A_test)*100
```

```
97.59223300970874
```

Figure 5.50 Training the First 250 Important Features

After many trials, we found that training the first 175 important features yields the highest accuracy. To obtain the names of the first 175 important features that have the highest scores, we used ". index [0:175]" and assigned it to a new variable called feature, as shown in Figure 5.51.

```
feature=feature_imp.index[0:175]
```

```
feature
```

```
Index(['tGravityAcc-mean()-X', 'angle(X,gravityMean)', 'tGravityAcc-min()-X',
       'tGravityAcc-max()-Y', 'tGravityAcc-max()-X', 'tGravityAcc-min()-Y',
       'angle(Y,gravityMean)', 'fBodyAccJerk-bandsEnergy()-1,16',
       'fBodyAccJerk-bandsEnergy()-1,8', 'tGravityAcc-mean()-Y',
       ...
       'tBodyGyro-std()-Z', 'tBodyGyro-iqr()-Z', 'tBodyGyro-mad()-Y',
       'fBodyAcc-std()-Y', 'tBodyAcc-correlation()-X,Z',
       'fBodyBodyGyroMag-meanFreq()', 'tBodyAcc-entropy()-X',
       'fBodyAcc-max()-Y', 'tBodyGyro-energy()-Z',
       'tBodyAccJerk-correlation()-X,Y'],
      dtype='object', length=175)
```

Figure 5.51 Training the First 175 Important Features

After assigning the first 175 important features columns names into the feature variable, we initialized "B" variable with the data from the most important columns only "data[feature]", and initialized "A" variable with the Activity column "data['Activity']". Then we split B and A data sets into test set (30%) and train test (70%). We initialized a new model with the random forest classifier and fitted it we the train sets "B_train, A_train", to store the machine prediction of the "B_test" data we initialized a "pred" variable, and then calculated the accuracy with ".score" function, As shown in Figure 5.52.

```
B =data[feature]
A =data['Activity']
```

```
B_train, B_test , A_train , A_test =train_test_split(B,A,random_state=42)
```

```
model1=RandomForestClassifier()
```

```
model1.fit(B_train,A_train)
```

```
RandomForestClassifier()
```

```
pred=model1.predict(B_test)
```

```
model1.score(B_test,A_test)*100
```

```
98.2135922330097
```

Figure 5.52 Finding the Accuracy of the First 175 Important Features

CHAPTER 6 EVALUATION

This chapter contains detailed information about the statistical evaluation performance and problems of our project.

6.1 Performance

By making use of our codes and the results in the previous section, it is seen that the most accurate method is determined as SVM. We supported this using the f-score. It is a statistical data used to decide which model should be the most accurate in data science projects. If we only choose the model based on Accuracy in our project outputs, we may get worse results that we should not only look at this metric.

6.2 Accuracy Calculation

The Confusion Matrix table we see below shows the actual and estimated values in a classification problem. True Positive and True Negative are the fields that the model predicts correctly, while False Positive and False Negative are the fields that the model predicts incorrectly as shown in Figure 6.1,6.2.

```
1                | Positive Prediction | Negative Prediction
2 Positive Class | True Positive (TP)  | False Negative (FN)
3 Negative Class | False Positive (FP) | True Negative (TN)
```

Figure 6.1 Separation of Class

Table 6.1 Meaning Of Classes

Key words	Meaning
True Positive	You guessed the store will churn, and it's true
True Negative	You guessed the store won't churn, and it's true
False Positive (Type 1 Error)	You guessed the store will churn, and it's wrong
False Negative (Type 2 Error)	You guessed the store will not churn, and this is incorrect

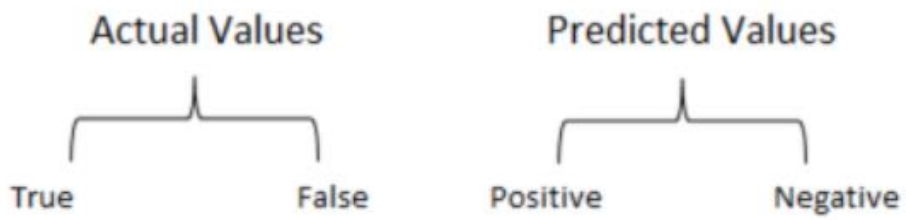


Figure 6.2 Separation of Values

6.2 F Score Calculation

Accuracy is a metric that is widely used to measure the success of a model but does not appear to be sufficient on its own. Accuracy value is calculated by the ratio of the areas that we predict correctly in the model to the total data set.Model accuracy alone is not sufficient, especially in unbiased datasets that are not evenly distributed.

$$\frac{TP + TN}{TP + FP + TN + FN}$$

Figure 6.3 Accuracy Mathematical Calculation

Precision, on the other hand, shows how many of the values we estimated as positive are actually positive as shown in Figure 6.4. The high precision value is an important criterion for machine learning model selection for us.

$$\textbf{Precision} = \frac{TP}{TP + FP}$$

Figure 6.4 Precision Mathematical Calculation

Recall, on the other hand, is a metric that shows how much of the operations we need to estimate as Positive, we estimate as Positive. If it is close to the predicted value, this brings us closer to the exact result as shown in Figure 6.5.

$$\text{Recall} = \frac{TP}{TP + FN}$$

Figure 6.5 Recall Mathematical Calculation

F1 Score value shows us the harmonic mean of Precision and Recall values, as shown in Figure 6.6. The reason why it is a harmonic mean instead of a simple mean is that we should not ignore the extreme cases.

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

Figure 6.6 F score Mathematical Calculation

	True Positive	True Negative
Predicted Positive	8865	426
Predicted Negative	63	646

Calculate

Measure	Value	Derivations
Sensitivity	0.9929	TPR = TP / (TP + FN)
Specificity	0.6026	SPC = TN / (FP + TN)
Precision	0.9541	PPV = TP / (TP + FP)
Negative Predictive Value	0.9111	NPV = TN / (TN + FN)
False Positive Rate	0.3974	FPR = FP / (FP + TN)
False Discovery Rate	0.0459	FDR = FP / (FP + TP)
False Negative Rate	0.0071	FNR = FN / (FN + TP)
Accuracy	0.9805	ACC = (TP + TN) / (P + N)
F1 Score	0.9765	F1 = 2TP / (2TP + FP + FN)
Matthews Correlation Coefficient	0.7179	TP*TN - FP*FN / sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN))

Figure 6.7 F-score for KNN

Measure	Value	Derivations
Sensitivity	0.9907	TPR = TP / (TP + FN)
Specificity	0.6141	SPC = TN / (FP + TN)
Precision	0.9562	PPV = TP / (TP + FP)
Negative Predictive Value	0.8861	NPV = TN / (TN + FN)
False Positive Rate	0.3859	FPR = FP / (FP + TN)
False Discovery Rate	0.0438	FDR = FP / (FP + TP)
False Negative Rate	0.0093	FNR = FN / (FN + TP)
Accuracy	0.9821	ACC = (TP + TN) / (P + N)
F1 Score	0.9802	F1 = 2TP / (2TP + FP + FN)
Matthews Correlation Coefficient	0.7138	TP*TN - FP*FN / sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN))

Figure 6.8 F-score for RF

Measure	Value	Derivations
Sensitivity	0.9841	TPR = TP / (TP + FN)
Specificity	0.6733	SPC = TN / (FP + TN)
Precision	0.9596	PPV = TP / (TP + FP)
Negative Predictive Value	0.8434	NPV = TN / (TN + FN)
False Positive Rate	0.3267	FPR = FP / (FP + TN)
False Discovery Rate	0.0404	FDR = FP / (FP + TP)
False Negative Rate	0.0159	FNR = FN / (FN + TP)
Accuracy	0.9915	ACC = (TP + TN) / (P + N)
F1 Score	0.9819	F1 = 2TP / (2TP + FP + FN)
Matthews Correlation Coefficient	0.7265	TP*TN - FP*FN / sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN))

Figure 6.9 F-score for SVM

```
from mlxtend.plotting import plot_confusion_matrix
import matplotlib.pyplot as plt
import numpy as np

binary1 = np.array([[8865, 426],
                    [63, 646]])

fig, ax = plot_confusion_matrix(conf_mat=binary1)
plt.show()
```

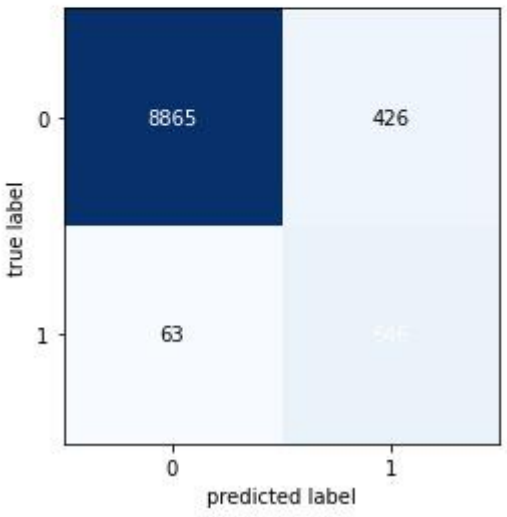


Figure 6.10 Confusion Matrix

After all these f score calculations as shown in the Figures 6.7,6.8,6.9, in the evaluation from 0 to 1, the method we approached the most to 1 was SVM with 0.9915. The parameters we used in our calculations differed, but the results that we determined by determining the best parameters are shown above. Our total evaluation results were calculated over 10000 data.

6.3 Problems

6.3.1 Finding Best Parameters

The most common problem we encountered while working on our project was finding the right parameters. It also took a long time to get the results of these parameters, the best parameters for all algorithms are shown in Figure 6.11,6.12,6.13.

```
knn_tuned = KNeighborsClassifier(n_neighbors = 4,
                                 metric= 'manhattan',
                                 weights = "distance")
```

Figure 6.11 kNN Best Parameters

```
svc_cv_model.best_params_
```

```
{'C': 30, 'degree': 6, 'kernel': 'poly'}
```

Figure 6.12 SVM Best Parameter

```
feature=feature_imp.index[0:175]
```

```
feature
```

```
Index(['tGravityAcc-mean()-X', 'angle(X,gravityMean)', 'tGravityAcc-min()-X',
       'tGravityAcc-max()-Y', 'tGravityAcc-max()-X', 'tGravityAcc-min()-Y',
       'angle(Y,gravityMean)', 'fBodyAccJerk-bandsEnergy()-1,16',
       'fBodyAccJerk-bandsEnergy()-1,8', 'tGravityAcc-mean()-Y',
       ...
       'tBodyGyro-std()-Z', 'tBodyGyro-iqr()-Z', 'tBodyGyro-mad()-Y',
       'fBodyAcc-std()-Y', 'tBodyAcc-correlation()-X,Z',
       'fBodyBodyGyroMag-meanFreq()', 'tBodyAcc-entropy()-X',
       'fBodyAcc-max()-Y', 'tBodyGyro-energy()-Z',
       'tBodyAccJerk-correlation()-X,Y'],
      dtype='object', length=175)
```

Figure 6.13 RF First 175 Important Features

6.3.2 Problem of Unbalanced Datasets

Unbalanced dataset is seen in classification problems and there have been cases where some of the datasets and class distributions we used were not close to each other. The problem is that the majority class dominates the minority class. The model created is close to the majority class, which causes the minority class to be poorly classified, so we spent a long time trying to find a more accurate dataset.

6.3.3 Accuracy Evaluation

It is seen that the number of classes in the data sets directly affects the results. Number of classes. As the number of classifications increases, the accuracy rate decreases. Trials show higher performance in datasets with less. Has been shown to yield positive results. Finally, the uneven distribution of the classes in the data set is also important in the evaluation of the accuracy value. Concluded that problems may arise. Using only a margin of accuracy may result in incorrect model selection. As a result, it is very important to choose the right classifier in classification studies and this It has been revealed that there are many parameters that should be considered in this regard. Specifically, the size of the dataset, the distribution of the classes, the number of classes and the test method to be used, together with the correct evaluation metrics. It is very important for a good classification to be used in a way that we have tried every method to increase the degree of accuracy.

CHAPTER 7 IMPACT OF THE PROJECT & COMPLIANCE WITH THE CONSTRAINTS

7.1 Compliance with the Realistic Constraints

Table 7.1 Realistic Constraints & Conditions

Economic Factors	**Please specify/explain realistic constraints and conditions (type, use, amount, etc.)**
EXPENDITURES	
Computer	We used our own computers.
Other Devices	NA
Peripherals	NA
Internet Connection	Currently owned connection was used.
Software	Anaconda Navigator, Jupyter Notebook, Python, Colabratory.
Textbook/Magazine/Support Material	
Human Resources	Our project team consist of 4 people.
Other	NA
FUNDING Sources	
University Resources	NA
Project support (SANTEZ, TÜBİTAK, and so on)	NA

Support by the Industry	NA
Self-funded	NA
Other sources	NA
OTHER CONSTRAINTS	
Memory	High memory RAMs are needed to work with large datasets.
Runtime efficiency	The size of the RAM memory used while training the dataset will provide efficiency.
MANAGERIAL	
Schedule (time)	During the project, it took us quite some time to train the dataset repeatedly and observe the results.

7.2 Impact of the Project

Our team evaluated the impact of this project and decided that it would be beneficial for data analysts working in the healthcare field. We evaluated the positive and negative aspects of the project. We evaluated that the aims and expectations of the system we developed would have a positive impact on data scientists in the field of health by exchanging ideas. The target audience of the project is data scientists working on nonlinear models. The main purpose of our project is to offer the best accuracy rates with the models used.

Table 7.2 Impact Assessment Report

Professional/Ethical Issues	**Please specify/explain (existence of items, violation of items, awareness about items)**

ETHICS/IT Law	Are there any applicable laws or legislation that will be relevant to the use and/or the construction of the system you are building? How have you addressed them?
Copyright in copying multimedia (sound, video, text)	Preparations are made for ethics.
Use of licensed software	Preparations are made for ethics.
Data Privacy	Preparations are made for ethics.
Use of patented products/ideas	Preparations are made for ethics.
IT Laws in Turkey (5661 and others)	Preparations are made for ethics.
IT Laws - International	Preparations are made for ethics.
Plagiarism	Preparations are made for ethics.
PROFESSIONAL	Liability/Sustainability Issues – What is the potential for liability either to yourself or your customer or users of the system, if it is misused or has flaws? Financial Impact – Have the financial costs of deploying and supporting the system been fully evaluated for yourself, the customer, users, and society as a whole?
Sustainability (use of Licensed and/or open source code)	We use open source application.
Maintenance	From time to time, the models can be retrained with new parameters added to the literature by conducting a literature search.

Liability	In case of a defect in the code, the responsibility belongs to the developer team.
Financial impact/Manufacturability	This system does not have any financials impact.
SOCIAL/POLITICAL/ ENVIRONMENTAL	Societal Issues – What is the potential for this system to be beneficial or detrimental to society as a whole? What have you done to address the potentially detrimental use? Are there any side-effects on the environment? Are there any potential political implications of the use of your system? What impact will this system have on the intended user community? Have you taken steps to safeguard the interests of that community?
Political impact	NA
Impact on health	The most detected activities can be determined and data can be used in the health sector.
Gambling	NA
Pornography	NA
Equal Access/equity	NA
Environmental impacts (energy, carbon footprint and so on)	NA
Technology acceptance & Human/Business psychology	NA
Security issues	NA

PROFESSIONAL (CODES, STANDARDS, FRAMEWORKS)	What technical standards are relevant to the software system you have built? How have you ensured conformance? These could be government standards, industry standards, or even just conventions that are followed in the market for your system. Be sure to clearly identify what type of standard you are talking about and who is the relevant authority for the standard.
IEEE	NA
ISO	NA
ANSI	NA
TSE	NA
ITIL/COBIT	NA
OTHER	APA for the preparing report.

CHAPTER 8 CONCLUSION

At the end of the activity classification project, we trained the UCI HAR dataset with 3 different nonlinear classification models. With metrics, it is possible to infer the prediction rate of each model and how each model responds to the dataset. With the GridSearch CV method, the accuracy of each model has been increased. The dataset modeled with Random Forest has an accuracy rate of 98.23%, with kNN 98.05% and with SVM 99.15% accuracy. As a result, it is seen that the model with an accuracy rate of 99.15% learns the mentioned 6 classes better than other models.

REFERENCES

[1]https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones

[2] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, *A Public Domain Dataset for Human Activity Recognition Using Smartphones*. .

[3] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "Energy Efficient Smartphone-Based Activity Recognition using Fixed-Point Arithmetic."

[4] Cost S, Salzberg S. A weighted nearest neighbor algorithm for learning with symbolic features. Machine Learning

[5] Guido S., Müller A. (2016, October), Introduction to Machine Learning with Python, A Guide for scikit-learn, 123-145

[6] HanJ., Kamber, M.(2006). Data Mining: Concepts and Techniques. Second Edition, 78-35

[7] G. Qu., X. Su and B. Han, "Residual Error Analysis of GPS Data Sequence based on WP", *Proceedings of 8th International Symposium on Spatial Accuracy Assessment in Natural Resources and Environmental Sciences*, pp. 22-29, Shanghai, China, 2008.

[8] Incel O D, Kose M, ErsoyIncel C. A Review and Taxonomy of Activity Recognition on Mobile Phones. Springer BioNanoScience Journal, 2013, 3(2): 145-171
Android Sensor gets user's moving direction (compass principle).(2015-12-21)
The explanation of how to transplant Tensor Flow training model to Android. (2017-2-20).

[9] Şadi Evren ŞEKER (2015,April 9). Ogrenme Egrisi https://mis.sadievrenseker.com/2015/04/ogrenme-egrisi-learning-curve/

[10]https://drive.google.com/file/d/1f7EBCsw4OU8E8larFl8XqMWpL6N7f4Sf/view?usp=sharing

[11] Fawwad Hassan Jaskanı (2020,May 5). Human Activity Recognition https://www.kaggle.com/code/favadhassanjaskani/human-activity-recognition-using-svm-knn/data