Learning Exergame Enjoyment

A Major Qualifying Project Submitted to the faculty of

Worcester Polytechnic Institute

In partial fulfillment of the requirements for the

Degree in Bachelor of Science in

Computer Science

On March 22, 2018

By Cameron Maitland

Submitted to: Professor Emmanuel Agu

____

Abstract

This Major Qualifying Project used data collection on how study participants played Pokémon Go to perform analysis and classify the enjoyment of Exergames. Using an Android app developed for this project, data about how users played the game was gathered and then uploaded for analysis. Descriptive statistics and analyses were performed using Plot.ly, while classification and classification result analysis were performed with MATLAB.

Table of Contents

1 Introduction

1.1 The Importance of Physical Activity

Physical inactivity and a lack of exercise have become increasingly prevalent in modern societies. Inactivity has been shown to be linked to many conditions, including obesity, cardiovascular disease and some cancers, and is a major contributing factor to many of the major health concerns in the United States. It plays a role in heart disease, high blood pressure, diabetes, metabolic syndrome and stroke. [1].

1.2 Lack of Exercise in the United States

According to the Centers for Disease Control and Prevention, less than 52 percent of Americans aged 18 and over currently meet the government's physical activity guidelines for aerobic physical activity, meaning they don't get the required 150 minutes of moderate exercise (brisk walking) and 75 minutes of intense exercise (swimming or jogging) recommended in a week. Even worse, only 22 percent of the same group meet the physical activity guidelines for both aerobic and muscle-strengthening activity [2]. Statistics for the youth population in this country are not any better. The latest National Youth Fitness Survey, which took place in 2012, had alarming findings for the rate of physical activity in the U.S for youth between the ages of 12-15. It found that only 25 percent of this population performed moderate-to-vigorous physical activity for at least 60 minutes each day [3].

1.3 First Console Exergames Promote Physical Activity

With physical activity decreasing, and the use of technology for fun and entertainment increasing, exergames present an opportunity to address this problem in a unique way. Since their origin, video games have typically been a sedentary activity. However, there are many games — exergames — that require movement and physical activity

Exergames are one way in which game developers have changed and are still changing the video game entertainment industry from an inactive one to one which puts the player in the position to get some much-needed exercise. The American College on Sports Medicine uses the following definition when classifying an activity as an exergame:

Exergaming is defined as technology-driven physical activities, such as video game play, that requires participants to be physically active or exercise in order to play the game.These game based physical activities go beyond simple hand finger movements as the primary interface and require the user to apply full body motion to participate in virtual sports, in group fitness exercise or other interactive physical activities. [4]

A well-known example of an exergame is the Nintendo Wii. The Wii, one of the first platforms designed to promote the playing of exergames, was released over 10 years ago, in 2006. It featured a unique remote controller that could detect players' movements. For the first several years after its release, many games were developed for the console, such as *Wii Sports*, *Wii Fitness*, and *Just Dance*, all of which were widely popular and got players moving and off of the couch.

1.4 Declining Interest in Console Exergames

However, after several years of record breaking Wii sales, Nintendo announced that it was stopping production of the console in 2013. Wii sold over 100 million units, appealing to a broad audience, ranging from children to families to seniors. No exergame, or video game for that matter, had appealed previously to these different demographic groups. In a last, failed attempt to reinvigorate the brand, Nintendo launched the Wii U in 2011. Third party support weakened, placing Nintendo in an impossible position. With few new games for the console, sales of the consoles plummeted. The end of a gaming phenomenon was over.

In addition to weakened third party support, there was another underlying problem affecting the Wii's popularity in the long run. Games such as those developed for the Wii require a level of activity and stimulation that many users could not keep up with. Games such as *Just Dance Two* require the ability to jump, process moving components on a screen in different colors, and perform precise steps [5]. Many users did not have the ability to continuously play these high intensity games, and so other alternatives became popular.

1.5 Exergames Move to Mobile Devices

As time has gone on and mobile smart devices have become commonplace, the mobile platform has started to support its own exergames. In recent years, mobile exergame applications have increased. In 2016, *Pokémon GO* was announced and it became a huge success. The game brought together the concepts of exergaming and augmented reality and the public loved it. To play the game, players interact with the real world. *Pokémon Go* had millions of people run around with their mobile devices outside, trying to find and capture the characters that are part of the game [6]. Even in April of 2017, almost a year after its release, Forbes reported that it still had nearly 65 million monthly users [7].

1.6 Geocaching and Exergame Enjoyment

A different type of exergame that has experienced a rapid increase in popularity are those that feature Geocaching. Geocaching is typically an outdoor activity that has players use some type of GPS receiver (the mobile device) to play a game resembling a blend of hide-and-seek and treasure hunting. Items are placed in specific locations, are marked by GPS coordinates, and then players try to find them. Geocaching in exergames is an example of location-based games that try to move the players from behind a screen into the real world [8]. In the Geocaching Motivations IQP, it was found that people geocached for multiple reasons. These include things such as, friendly competition, exercise, and being able to explore outdoors [9].

Geocaching has proved to be an entertaining form of exercise that also provides players with other benefits. The U.S. National Institute of Health conducted a study of geocaching in 2015[10]. It interviewed participants and asked them questions about how and why they use geocaching. The study found that most had geocached for more than five years and had done so with a consistent frequency, at least once a week. The reasons given for their interest in geocaching as recreation were “being outdoors, social interaction, physical activity, and relaxation.” Individuals described geocaching as “being part of a community.” Players made friends and felt a connection with other players, because of their shared interest [10].

1.7 Why do Players Quit Exergames?

There are numerous reasons for people to want to play exergames. These range from the exercise you get from it, to the friendly competition you can create with it. However, even though exergames have been shown to provide a multitude of benefits, players often quit these games prematurely and without warning, nearly 85% of players quitting after a day, and 95% within 3 months [1]. This is puzzling. Why does this happen and how can exergames be designed for long term enjoyment?

1.8 The Goal of this MQP

Technology has become more and more pervasive throughout the past decades, and continues to do so. With the advent of wearables and the increasing prevalence of smartphones, it becomes more and more possible for us to gather data on how people are playing games. Through studying these games, we can utilize machine learning to look at the data collected. Here, we can attempt to make sense of the data collected. In this projects example, we can attempt to have participants play exergames with their apps, and determine whether or not they are enjoying the game. With this information, we could proceed in different ways, recommending different types of games if they aren't enjoying it, and recommending similar games if they do.

The goal of this project is to look at these questions and determine what traits can be used to maximize the long-term enjoyment of exergames by their users. This project's focus will be to:

- Develop a list of potential features that will help determine a player's level of exergame enjoyment.

- Develop and deploy a test application in a study to see how well we can collect these features.
- Utilize classifiers to calculate and quantify the player's enjoyment.

2 Background

2.1 History of Mobile Games

In 1972, the Magnavox Odyssey was released. The Magnavox unit was the first video game console designed for use in the home environment. One of the 28 games made for the Magnavox system was a ping pong game. This game inspired the creation of *Pong*, which became one of the first popular arcade games from Atari. *Pong's* popularity circled back to feed the popularity of the Magnavox console.

The success of the Odyssey unit showcased the potential that video games had in the technology industry as household products. Since the early days of video games, users from around the globe have been drawn to play. Players range in age from young to old and are both male and female. The average age of those playing these games ranges from 18 to 49, and even though the stereotypical gamer might be considered by many to be male, the number of females playing these games is almost equal to the number of males, according to the Entertainment Software Rating Board(ESRB) in 2015 [11]. Also, according to the ESRB, the number of females playing video games continues to grow.

These statistics show that although the video game industry started out as a niche form of entertainment, the industry has boomed into a pervasive part of our culture, with more and more diverse groups of people participating. Initially, the industry was based on arcade game systems. With these, companies could expect to sell hundreds of thousands of copies that would be shared in public spaces. With the success of Magnavox, developers started to realize that video games appealed to the masses and had the potential to sell millions of consoles for home entertainment [12].

In the past, to play a game such as Pong, players needed to purchase a console and the game cartridges or CD's. In modern times, however, games are always "an arm's length away" [11]. As mobile phones have grown in popularity and are now a standard fixture in our everyday lives, the potential for users to play an electronic game without the need to purchase additional hardware, and for a fraction of the cost, exists. The electronic gaming market has exploded and, today, users can simply use their mobile phones to access the application hub and find thousands of free games to play. Electronic games are now accessible 24/7, right at a player's fingertips.

Within the last several years, a majority of the most popular video games have been designed for mobile devices. Games such as *Angry Birds*, *Clash of Clans*, and *Pokémon Go* have been downloaded by millions of users from all over the world. With their non-existent starting cost, users can quickly and efficiently get into a game and begin enjoying it. This easy access has helped make these games very popular. The most popular games have even become cultural phenomenon's.

The accessibility of mobile games has brought new users from parts of the world that console games simply were not designed to reach [11]. Real world features, such as locations, are largely unattainable or irrelevant in console video games. Location-based exergames games such as *Ingress*, which also uses augmented reality, was released in 2012 by Niantic Labs, a company that was spun-off from Google. Games such as *Ingress* allow users to be engaged in a game on their phone, which they already have on their person a majority of the time, while also allowing the user to interact with the real world as part of the game.

2.2 History of Exergames in the Video Game Industry

The origin of exergames can be traced back to the release of the Joyboard for the Atari 2600 in 1980[13]. It is interesting to note that during this same year, Atari planned the release of the Atari Puffer, which would have been one of the first examples of an exergame in the video game industry. Sadly though, the system wasn't released due to financial reasons [13].

The Atari Puffer featured a more simplistic design for exergame systems than what we see now. The entirety of the system was designed to be essentially a stationary bicycle. The bike's handles had buttons on them to control navigation and the wheel would control how fast the user would go in the game [14]. Developers came to realize the best types of games for this system were those where pedaling and braking were the primary focus of the game.

The Atari Puffer
https://kotaku.com/5009577/atari-puffer-the-wii-fit-of-1982

Games such as *Jungle River Cruise* and *Tumbleweeds* demonstrate how this technology was featured [14]. *Jungle River Cruise* turned out to be a largely successful test, as the controlling actions on the bicycle were very intuitive. Users would pedal and brake to avoid crashing into obstacles while going down a river. In contrast, *Tumbleweeds* had a more difficult time. The games features were similar to *Asteroids*, and users found the controls non-intuitive, leading to numerous playing issues [14]. Developers learned that simple intuitive game design was necessary for mass appeal.

Several years after the bankruptcy of Atari, Nintendo came out with the next innovation in exergames, the Power Pad. In 1988, this mattress-like invention contained twelve sensors [13]. These sensors were activated when users stepped on them. Soon after its launch, the game *Dance Aerobics* released, utilizing the same type of sensor, becoming one of the first exergames to make it to a commercial stage [13]. The Power Pad set off the dancing game revolution of the 90’s with games such as *Dance Dance Revolution* being introduced [13]. This series of games remained highly popular throughout the early 2000’s and continues to be played today. The *Dance Dance Revolution* product group has been hugely successful and has had staying power, much more than most other exergames up until now. There have been many arcade releases, as well as home video game console releases, over the years, with a new release as recent as 2016.

The Nintendo Power Pad

https://www.amazon.com/Nintendo-NES-Power-Pad-Entertainment-System/dp/B002W4JN9Q/ref=sr_1_1?ie=UTF8&qid=1521765402&sr=8-1&keywords=nintendo+power+pad

From the early 2000’s to now, the popularity of exergames has grown tremendously. In 2006, Nintendo launched the Wii console, which experienced huge success. The Wii launched in November 2006 and by the following August it had sold over 250,000 units, eclipsing game systems like Sony’s PlayStation 3’s 80,000-unit sales that same month [15].

In the years to follow, more and more consoles have been released, such as the Wii U, PlayStation Move, and Xbox Kinect. As time has gone on, this type of platform has been extremely popular, promoting exercise and prompting users to get up and off of their couches along the way. While none of these systems became the sensation that the original Wii was, they have sold millions of units and made their companies a lot of money by continuously improving their features and, thus, the user experience.

The most recent frontiers of these games include mobile devices. In 2016, Niantic launched *Pokémon GO*. This exergame combined geocaching and augmented reality and turned into an overnight sensation. *Pokémon GO* seemed to become a seemingly overnight sensation and created a media frenzy. The popularity of this game was enough to crush the networks it was running on. While the system was designed to support an anticipated four and a half million users, the game was estimated to have peaked at over 45 million daily users [16].

2.3 Android Development

Android is the operating system market leader in mobile devices. In the third quarter of 2016, Android reached a milestone by capturing a whopping 88% of the worldwide market, according to Strategy Analytics. During the same time frame, Apple’s iOS operating system fell from a 13.6% market share to 12.1% compared to the prior year. What is behind Android’s popularity and will it continue? In figure 2.3.1 it can be seen how Android's market share has been on a slight rise since 2013. While apple has started to show losses in market share, Android has continued to rise, never decreasing since 2014.

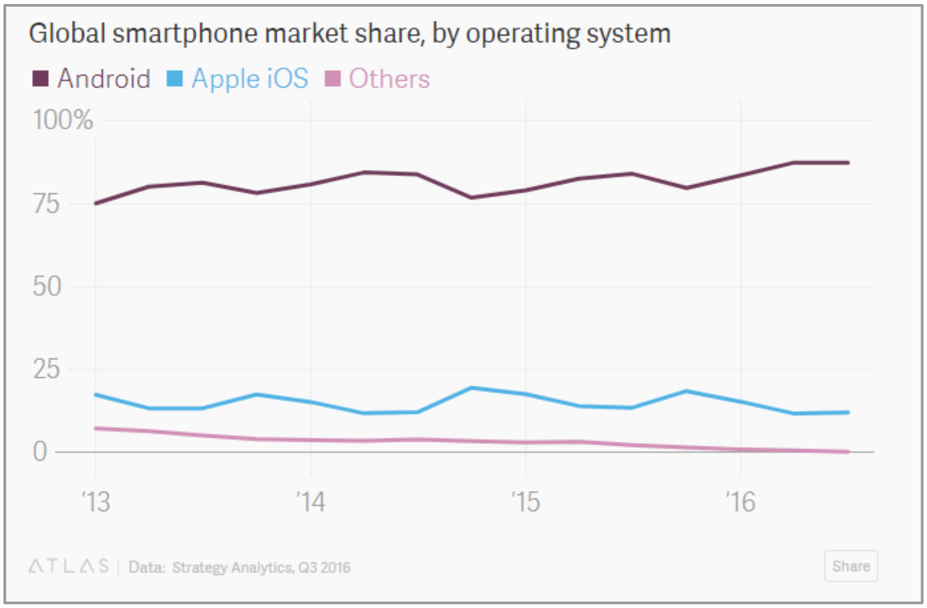


Figure 2.3.1: Smartphone Market Share [17]

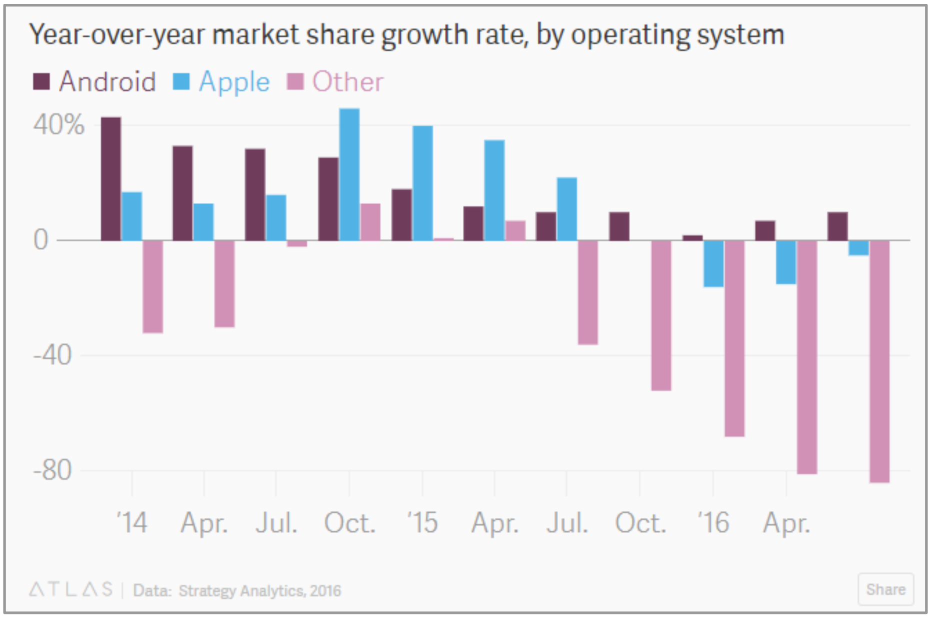


Figure 2.3.2: Market Share Growth Rate [17]

The main reason Android has been so successful is that it partners with many companies that are continuously releasing new phones at a variety of relatively high and low-price points. This explains why Android practically owns the market in developing countries, such as India, Africa and the Middle East, where many consumers cannot afford the latest iPhone. While Android owns 97% of the market share in India today, just seven years ago it had less than five percent of the global market. Android's success is coming at the expense of its competitors, both new and old, such as the Windows phone and the Blackberry. Consumers who are "moving from declining platforms are mostly moving toward Android," says Neil Mawston, executive director of Wireless Device Strategies (WDS) at Strategy Analytics [17].

Android faces major challenges as it continues to grow. The biggest challenge is the issue of fragmentation. Android developers have to create software that is compatible with many different devices from dozens of manufacturers. This presents a multitude of complications that Apple does not have to work through. Another challenge is that Android is open source. This means that manufacturers can revise and adjust code as they see fit. This leads to the fragmentation problem mentioned above, as well as a problem with updates, since Google can't push Android to devices because of modifications that might have been made.

What does the future hold for Android? No one is sure, but enhancements will surely be developed. Late in 2016, rumors started to circulate that Google might be working on a future Android operating system that would solve the fragmentation issue and be able to operate effortlessly across all devices. It would also be able to get regular updates, as the iPhone does. However, one of the first projects to address this issue, the Andromeda project, was scrapped in the summer of 2017. Currently, a separate project, Fuchsia, is being talked about. Fuchsia supposedly won't be a hybrid operating system to bring existing mobile and desktop efforts together, but a completely brand-new platform that would function across devices from phones to tablets and laptops. Even though Fuchsia is not certain, one can be certain that new developments for the Android operating system will be coming [18].

2.4 Phone Sensors

Today’s mobile phones are equipped with sensors to automate many functions to provide a satisfying user experience. Sensors can identify and track just about anything that is physical in nature. They are built into the phone and work in the background, without the user being aware of them. Examples of sensors present in most mobile phones are:

- Accelerometer - detects the movement of the phone and its orientation, which enables features such as changing music by shaking the phone
- Ambient light sensor - extends battery life of the phone by adjusting the screen’s brightness based on the level of available light
- Barometer - measures air pressure and provides altitude data that helps the phone’s GPS chip identify a location faster
- Digital compass - locates the North direction, to enable mapping and navigation applications to function
- Global Positioning System - connects with multiple satellites and calculates the phone’s location based on the angles of intersection
- Gyroscope - works along with the accelerometer to detect rotation of the phone, which is needed for features such as tilting the phone to play certain games
- Image sensor - takes photos using technology that produces quality images with smaller pixels

- Magnetometer - measures magnetic fields and works with the digital compass to identify North and which way the map should be oriented on the phone
- Proximity sensor - locks the screen so that accidental touch commands are disabled when the phone comes close to the ears during a call

Sensors have become a more and more relevant way of having a phone interact with the world around us. In the past, phones were limited in the interactions they could have with the world, being unsure of the conditions their users were facing. As sensor technology continues to develop, more and more exciting things will be able to be accomplished with our mobile devices.

2.3.1 Android Sensors

Android provides a plethora of phone sensors that are available to developers. Sensors are used to locate and track just about anything that is physical in nature. Using these phone sensors, developers are able to process what is happening in the world around the end user, and try to more accurately create a picture of what has, is, or will happen. Android sensors available on the Google Pixel 2 include: [19]

- Active Edge™
- Proximity / Ambient light sensor
- Accelerometer / Gyrometer
- Magnetometer
- Pixel Imprint: Back-mounted fingerprint sensor for fast unlocking
- Barometer
- Hall effect sensor
- Android Sensor Hub
- Advanced x-axis haptics for sharper/defined response

Developers must deal with the challenges presented by Android's sensors, since different phone models present different sensors. Phones such as the Google Pixel 2 feature Android's sensor hub, a secondary processing unit. This processor performs low level operations, allowing developers to get data much more easily. Things such as step count or which activity (running, jogging, or walking) the user is doing, are computed on the chip from other sensors. However, in other older devices, some of the features found in the Android sensor hub are not present and must be calculated from other sensor data.

This project deals with this issue, because the application includes a user's step count. Phones such as the Samsung Galaxy S5 and the Nexus 5 both have this sensor available, but users on older phones do not have these sensors available.

Although step count is accepted as a sensor and is used as such, it is not a raw sensor, such as the accelerometer and barometer. An accelerometer in a mobile device tracks and measures movement. In contrast, this sensor relies on the phone having the Android sensor hub present. On the Android sensor hub, the step count is computed from this data, however, without it, the data must be calculated externally through machine learning using data from another source, such as the accelerometer.

In addition to the difference in sensors available on phones, the way developers use Android sensors have changed drastically over time. During the early days of Android sensor programming, calls were made specifically to the driver for the necessary sensor. Now, users make a call to a sensor Manager, where users interact with sensors in one standard way, and just state which sensors they want to utilize.

Because of these issues, there are currently multiple ways to get sensor data through your phone and third-party Application Programming Interfaces (API's). Sensor features such as step count have been integrated into Android devices running with sensor hub hardware, as well as third party API's such as Fitbit, Jawbone's UP, and Misfit. Utilizing these, developers can get a better understanding of the characteristics of the environment their users are exposed to, as well as the users themselves.

2.3.2 Google Fit

Google Fit is an app which allows users to monitor their physical wellbeing. Alongside the app, Google introduced an API of the same name which allows developers to monitor the features the user grants permission to. Things such as how long you've been active in a day, how much you've walked, and weight tracking are all available through the API and app.

Google Fit primarily operates in two ways, via REST and Android API calls. REST stands for "Representational State Transfer." REST is an architecture for designing networked applications. It utilizes a client-server, cacheable communications protocol, which is usually HTTP.

The point of Google Fit is to attempt to bridge the hardware limitations of the various Android phones. Since some devices do not contain the hardware necessary for certain sensors such as step count, applications had difficulty dealing with a wide variety of devices. An application made with the pure Android sensor would work on one phone, and then another version that utilized machine learning would be required on another phone.

2.3.3 Shimmer

Open mHealth describes Shimmer in this way: "Shimmer is an application that makes it easy to pull health data from popular third-party APIs like Runkeeper and Fitbit. It converts that data into an Open mHealth compliant format, letting your application work with clean and clinically meaningful data.” [20]

Shimmer currently supports the following APIs: Fitbit, iHealth, Jawbone UP, Misfit, Moves, Runkeeper and Withings and has announced plans to work with FatSecret, Ginsberg, iHealth and Strava in the future. It is made up of three different components: shims, a resource server and a console. Shims are libraries that interface with one of the third-party API’s for authentication and mapping purposes, ultimately generating data points. The resource server allows enables the API to retrieve the data points. The server handles API requests by assigning them to the correct shim. The console provides a simple web interface that helps users interact with the resource server. It is necessary for authentication and requests data using date pickers and drop downs.

With Shimmer, developers can consolidate various API’s, from fitbit to Google Fit to iHealth, into a single REST API and have the server handle the authentication for each API. From here, the data is stored into a single normalized format, as opposed to multiple conflicting formats. This allows developers to more rapidly create software without having to worry about various API overheads.

The upside to this system is that it greatly speeds up the development process and allows for a more diverse group of subjects to survey. When looking at standard sensor calls, mobile applications such as iHealth or Android’s built in sensor API require that the user be on a specific type of hardware. When using Shimmer, this becomes less important.

It is worth noting that although Shimmer is a useful tool for collecting data from a large variety of devices, the devices themselves may not support the same types of information. For example, Google Fit, which is present on Android by default, utilizes sensors available via

Open mHealth’s API, including geographic position, step count, speed, activity, and body weight. Meanwhile, Fitbit supports sleep duration, sleep episode, step count, and physical activity. While there is overlap in what is available to users, the system cannot pull everything from all API’s. To illustrate this further, a user with a Fitbit, but not Google Fit, can access sleep information, but won’t be able to get location information, while a user with Google Fit, but not a Fitbit won’t be able to get sleep information.

In the end it was decided not to use shimmer. The reasoning for this, is that the target audience for the study was Android devices, and step count was the only health feature used. The point in time I believe it would become practical to start using Shimmer would be when you start targeting alternative phone platforms such as IOS, or wish to implement other alternatives for data collection such as Fitbits wearables.

2.5 Drobox

Dropbox, which launched in 2007, is a platform that stores files and automatically keeps files synced and backed up online. Like Google Drive and One Drive, it is a standard storage system with an easy-to-use API. It creates a folder on the user’s computer and then synchronizes the contents to its servers and other devices. Dropbox provides several services: storage in the cloud, automatic file synchronization across all devices, and a personal cloud The Drobox app is available for Windows, Apple OS and Linux operating systems. It offers a mobile app for Windows mobile devices, iOS and Android devices. Dropbox offers free service with 2 gigabytes of storage or a premium version with 1 terabyte of storage and additional features for $9.99 monthly. Currently it has over 500 million customers.

For this project, Dropbox will be utilized as the means of remote file storage. The app will create .csv files containing all of the sensor data collected and send it to Dropbox every day.

The alternative to cloud storage such as Dropbox would be dedicated storage. The disadvantage of this, is that it is much harder to scale. When dealing with small amounts of data space such as happened in this study, it is unneeded to have a large Database to store the data. Similarly, the system is much simpler to set up for simple file storage.

Even as scale continues to increase, it is good to have the flexibility and reliability of a third party such as Dropbox to keep track of files. If you get to the point at which you end up using a server to manage data, you can still use Dropbox to handle files created by a phone, and use the server for data processing as opposed to storage.

2.6 Machine Learning

As humans, much of how we process information and learn comes from our prior knowledge and experiences. As we move forward, we look at the past and create new ideas and expectations about what happens around us. This is also what machine learning seeks to accomplish. Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed [21]. Through machine learning, computer programs access data and use it to learn for themselves. Statistics and Statistical Learning, Pattern Recognition, Signal and Image Processing and Analysis, Computer Science, Data Mining, Machine Vision, Bioinformatics, Industrial Automation, and Computer-Aided Medical Diagnosis are all well-established fields which have used similar techniques for many years [22].

Today’s machine learning has evolved to where it can automatically apply sophisticated mathematical calculations to big data. In recent years, the general population has come to enjoy its capabilities in applications like the online recommendations provided by Amazon and Netflix, the self-driving car and fraud detection programs. These are all everyday applications of machine learning.

The role of machine learning in exergames is to use the data collected by the movements of the user via the smartphone’s sensors to determine the enjoyment of the game. By utilizing these techniques, one can attempt to classify a user's enjoyment of an activity.

Much of what this project seeks to accomplish falls under the category of game analytics. A big part of the video game production process is looking into popular trends in video games and then analyzing their player retention rate and how players interact with them. Much of what this project seeks to accomplish is closely related to published papers about how to retain users in other genres of video games. This project’s methodology is relatively similar; however, some of the data, as well as the data collection methods, will be different.

One example of a work that had a similar objective was “Predicting Disengagement in Free-to-Play Games with Highly Biased Data.” [23] This paper sought to do something similar to us. Just as exergames have a high turnover rate in participation, so to do free-to-play games. Here, the paper’s authors looked at variables such as the frequency of playing games, how often players participated with others, and how often players spent money on the app, among other things. [23].

They used the data they collected to run multiple models, such as decision trees, vector machines, and logistic regression. [23] They chose to name the approach they utilized “trend over varying dates.” This was a methodology they took to create classes of data as accurately as possible, improving over time as the study progresses. Part of what they found was that to maintain a balanced distribution of classes, rather than using explicit values for cut offs, it was more accurate to use a function that varied the cutoffs as time went on.

This paper served as inspiration for this project’s design and methodology. Although this project will look at different data, the overarching concept behind why the data is being collected and how it will be analyzed is like what is described in the paper, but with different data points. Features that are relevant to look at, such as if a player expands the game by inviting more people and others noted above, are examples of data points that this study will also examine.

3 Methodology

3.1 Study Design Summary

To begin the study, participants were required to download an application called ExergameMonitor developed for this study and available via the Google Play Store. Using the email provided, a link was sent out to users with the option to opt in to a beta for the application on the Google Play Store for Android phones. From here, users downloaded the application on their phone through the Play Store.

For this study, users played the game for one week. At the beginning of the week, users were asked to fill out a pre-study questionnaire. This contained information such as the participant's email, their experience level with exercise games, whether they enjoyed them or not, how much they've played them, and whether they've played a game utilizing Geocaching before.

After filling out this form, the user played the exergame for one week. During this time, the user had the ExergameMonitor running. While doing so, the application collected GPS locations the player frequented and locations they visited while playing the exergame. It also recorded the temperature every hour, as well as the step count, while they played the exergame.

In addition, users were asked to fill out the Exergame Enjoyment Qualifier once a day. In this daily survey, players were asked to rate their experience playing the exergame on that day. This survey featured a Likert- based scale, where values are weighted with a score of 1 to 5, with one being "strongly disagree" and 5 being "strongly agree." A week after the initial survey was completed, the application communicated that the study was complete. After this point, machine learning was employed using classifiers to evaluate the enjoyment level of a participant.

3.2 Pokémon GO

Pokémon Go was the game that was determined to be best for this study. The reason for choosing this as opposed to Geocaching, is that it had a much more mainstream appeal due to its peak in popularity. Because of this, it was easier to find participants who would sign up to play a game they'd heard of before. In addition, more people have played the game before, and there are more resources for those who need help with the game. This meant I could focus more so on the data collection and analysis, and less time working out a participant's logistics with playing the game.

In Pokémon Go, the user travels around the real world with their phone. Certain locations in the real world correspond to even triggers in the game. As you approach a location various events occur from going to catch a Pokémon by swiping to throw a poke ball at it, to getting to a poke stop where you can stock up on resources such as poke balls and eggs. While playing, the player builds up better and better resources through the acquisition of things such as better poke balls, and the user seeks to raise the level and as collect as many Pokémon as they can.

3.3 Data Collection

3.3.1 Data to Collect

Three pieces of data were collected on the user's phone of the course of the study. From this data, multiple features can be derived. The first piece of data was gaps data. This was used to calculate where the user was for weather information, as well as movement statistics in the game. The second piece of data was weather data collected from OpenWeatherMap, this returns numerous features such as wind, temperature, and rainfall. These were important pieces of information, because when talking about outdoor activities, weather can largely influence what happens. Precipitation can be reason for event cancellations and humidity and temperature are things which can highly effect how one exercises. The third piece of data was step count, provided by GoogleFit, and the final data stream was the times Pokémon Go was opened.

For this study, several features were identified that were believed to be relevant in determining if the player would enjoy the game and the level of his or her engagement with the game. These features largely concern the conditions participants are willing to endure while playing a game, as well as how frequently they are willing to undergo these various conditions. These features include:

1. Step count while playing the exergame
2. Step count the day of the exergame
3. Average temperature the day of the exergame
4. Average humidity level on the day of the exergame
5. Average precipitation on the day of the exergame
6. Average wind conditions on the day of the exergame
7. Earliest time the application was opened
8. Latest time the application was opened
9. Average lag time between exergame sessions
10. Number of times the application was started
11. Distance traveled while playing the exergame
12. Furthest distance from home when playing the exergame
13. Average length of session

14. Prior experience playing the exergame (response on a Likert scale)

15. Total time the exergame was played on that day

Although some of these features were raw system data, others are items that required extrapolation from the device. For example, step count is created by making an API call. For distance from home, the decision had to be made “what exactly is home” and then proceed from there. Distance from home was defined by the user. During the initial survey, the user was asked to provide an address to be used as the home location.

For data collection, the time a session started was defined as the time the application was first opened. The end time was a little less precise. When playing a game or using an application in general, users often minimize applications or put their phone to sleep while walking around. Because of this, data in this study continued to be collected until the user went into the application manager and exited out of the process. This study identified when the application was or wasn’t in the foreground, but continued to gather data either way until the application was closed.

3.3.2 Pre-Questionnaire

The pre-questionnaire was used to collect information’s on the demographics that would participate in the study. This asked which address they wanted to use so that I could establish their home location for GPS, as well as demographics questions. The questionnaire can be seen in figure 3.3.2.1.

What is your Gender?
What is your Age?
What address would you like to be your home?
How much exercise would you say you get in a typical week?
Have you ever played a video game before?
Have you ever played a game on your phone before?
Do you consider yourself as having played an exercise game before?
Have you played *PokemonGO* before?
If you have, how long did you play?

Figure 3.3.2.1: Pre-Questionnaire

3.3.3 Exergame Enjoyment Questionnaire

To have data to compare experiences across all players, participants in the study completed the Exergame Enjoyment Qualifier (EEQ) at the end of each day. This study asked the user about their experience with the exergame throughout the day. The study was developed during the *Run For Fun Measuring Exergame Enjoyment* IQP completed at WPI. This questionnaire is based on several Likert scales, such as the Immersive Experience, Game Engagement, and Physical Activity Enjoyment questionnaires [24]. Questions include things such as how excited you were about the exercise, and did the exercise make you feel good.

On this Likert type scale, each response is given a score of 1 to 5. On a question where a high agreeance would be a bad thing, a value of 1 would go to highly agree, while a score of 5 would go to disagree. On the opposite side, if something is beneficial and they highly agree, it would be scored as a 5, versus disgrace a 1. To figure out a response total EEQ, you add the sum of the values of the responses.

Question	Response Values
I felt excited about the physical activities in the game.	1-2-3-4-5
The exercise in this game made me feel good.	1-2-3-4-5
I felt like I lost track of time while playing	1-2-3-4-5
I felt that it was difficult to understand how the game works.	5-4-3-2-1
I was focused on the game.	1-2-3-4-5
I felt that the game would have been more enjoyable without physical activity.	5-4-3-2-1
I felt that it was easy to familiarize myself with the game controls.	1-2-3-4-5
I felt emotionally attached to the game.	1-2-3-4-5
I consider playing the game “exercise”.	1-2-3-4-5
I felt that the physical activity was too intense for me.	5-4-3-2-1
I did not feel a desire to make progress in the game.	5-4-3-2-1
I felt a strong sense of being in the world of the game to the point that I was unaware of my surroundings	1-2-3-4-5
I would rather not be exercising, even though the exercise was accompanied by game elements.	5-4-3-2-1
I felt that playing the game was beneficial for my physical well-being.	1-2-3-4-5
I felt that this game provided an enjoyable challenge.	1-2-3-4-5
I felt a sense of accomplishment from playing the game.	1-2-3-4-5
I felt that the game reacted quickly to my actions.	1-2-3-4-5

.I did not feel like I wanted to keep playing.	5-4-3-2-1
I would prefer that this physical activity was not accompanied by game elements.	5-4-3-2-1
I felt in control of the game.	1-2-3-4-5

Figure 3.3.3.1: Exergame Enjoyment Questionnaire

3.3.4 Data Collection Methodology

To collect the data, an application that monitored all this information was created. The application had two components: the user interface and a background service.

The user interface is where all the information the user needs to participate in, such as the surveys and the help page was stored. Here, there are several pages. When the user first downloaded the application, they were presented with a survey for basic information. This included information such as email address and questions involving their experience with exergames.

After the initial questionnaire, the application's user interface was broken up into several pages. The first was where the EEQ was located. Here, users could fill out the questionnaire. The second tab provided a help page, which contained information about what they were expected to do throughout the study, as well as further disclosure of the type of information being collected.

The background services recorded all this information, storing it and transferred it to Dropbox. Each service ran for a different data type required, collecting its own data. Every day at midnight, a final service that extracted all the data from the phones database and uploaded it to Dropbox.

3.4 Pilot Studies

3.4.1 Design Pilots

In the initial stages of planning the study, the primary focus was to decide what types of surveying techniques users would be comfortable with (how often they would agree to complete the survey, etc.). This included users' preferences on user interface design and how consistently they would answer surveys. If you ask the user to fill out a survey every time they open the app, its highly unlikely they will answer the survey every time, as well as sometimes the survey wouldn't provide much.

The first pilot study was to see how users reacted to different forms of installation of applications. The two methods the study asked them to try were to install the application Androsensor via an APK file emailed to them and via the Google Play Store beta program. Users strongly preferred downloading the application via the Google Play Store.

Because of this, the study used Google's dev console to distribute the application. With Google's dev console, a beta list containing email addresses was created from a .csv. After uploading this, a link was sent out to the email distribution list so that players who clicked the link were taken to the application under a beta section on the Google Play Store. The users then went to the beta section of the Play Store and simply downloaded the application from there and were provided with an email containing contact information in case any issues arose. As more people were added to the survey, we simply added them to the beta list.

During the study design process, a group of ten college students was asked about their survey preferences. They were asked how often they would fill out the questionnaire and whether they knew how to fully exit out of an application. They all knew how to exit out of an application fully. The consensus was that participants would reliably fill out the survey once a day. It was believed that asking for more than one survey per day to be completed would not produce accurate results, as participants would tire of the process and not spend time to think about their answers and answer questions accurately.

3.4.2 Data Collection Pilots

The goal of the second round of pilots was to test how data collection worked. There were two initial studies of the data: a one-day test to see how the data was being formatted (which files were being written, where were they written to, what was there format, how large did they get) and a second weeklong test to see if the data was being integrated with Dropbox properly.

For the first round, a college student used the application for one day. During this time, the only data that was collected by the application was the step count data. The data was used to test whether the application properly formatted in CSV and wrote to the same file in a consistent readable manner.

The second pilot study took place during a break in the fall. At this point in time, the application did not have all the data types present, but did have the storage and upload mechanism complete. The goal of this pilot was to determine if there was any difference in the way the data collection process worked for one day versus many.

3.5 Study Recruitment

For this study, the objective was to get 60 participants. Study recruitment came from two primary sources: friends, and online forums and communities. Using these different populations ensured a diverse, representative pool of people.

Both populations introduced participants from different backgrounds into the study, with different motivations for why they would play the game. Friends would be willing to go out of their way to help with the project. They most likely would play the game on their own time and possibly for a longer period of time than other participants and not be in a rush to finish it as quickly as possible. They might have limited knowledge of the game, but would earnestly give it a try. There was also diversity within the friend group. Some friends were less motivated to participate in the study, but agreed to because they were already playing the game as a company activity, and participating in the study was easy to do and simply expanded upon how they play the game.

Online forums were the more interesting group to ponder. Websites like Neoseeker and Gohub are filled with people who play the game regularly on their own time. Unlike the other two groups, these people are more diverse geographically and feature more experienced players. Other online forums used for recruitment were the WPI *Pokémon GO* group, as well as the Worcester *Pokémon GO* Discord server. Students seeking class credit on SONA were not selected for the study as their motivation might affect the study's results.

3.6 Analysis

The first type of analysis was done using Plot.ly [25]. Using this tool, data was uploaded into a spreadsheet (.csv) format and then analyzed. From here, analyses were generated, such as simple regression on data, mean, and maximums/minimums of data points. Using linear regression, data was analyzed to uncover trends. Because of the descriptive information, general traits contained in the data could be analyzed that would help to more accurately determine predictions of outcomes.

Plot.ly also facilitated the development of graphs. Different graphs were generated using scatter plots, box plots and contour plots. These different options allowed the data to be accurately represented in a formal, cohesive and visual manner.

The second type of analysis was focused on shallow learning. This was primarily done via MATLAB. Through MATLAB’s classifier learner, various classifiers were applied to the data. These included, among others, Naive Bayes classifiers and Decision Trees [26].

4 Implementation

4.1 Exergame Monitor Application Implementation

Exergame Monitor is the name of the application developed to collect the data required for all the features, as well as to upload to Dropbox. The application itself was written in Java for the Android platform. The application was developed using Android SDK version 26(Oreo), and was targeted at version 26, but will work with a minimum version of 19(KitKat).

The largest functional area of the application is the services. The application services are broken up into the four pieces of data that needed to be gathered (weather, play statistics, step count and GPS) and then a final service uploaded the files. How the data was collected varied from service to service. For location, Android's built-in GPS manager was utilized, as it is nearly ubiquitous throughout all Android devices now. Although this doesn’t work indoors, it was used because the assumption that they would not travel too much distance while playing indoors.

Game play statistics were found via the activity manager. Through here, statistics were available on which processes were currently being run on the phone. Game play statistics were implemented to gather data by scanning the phone every five seconds. If the application was within the activity manager’s list of processes, it was active on the phone. This then marked the time the user opened the application and wrote it to the game statistics csv. Once the application left the activity manager’s list, it was assumed the application had been closed and recorded the closure to csv.

Step count and the weather service worked differently from each other, but were both third party API calls and required a third-party authentication key.

In order to provide the step count, Google Fit was utilized. This is an API shipped inside Google Play services. Using Google Fit, calls were made within Android to Google’s packages and collected health statistics about the user, including step count.

The weather service was implemented with OpenWeatherMap’s REST API. With this, weather data is collected at a given time or recalled from its history for any given GPS location. This method was utilized, because it collected more weather data than Android sensors support. Data such as humidity, precipitation, and wind speeds are not available via Android sensors. Figure 4.1.1 shows the data from OpenWeatherMap used.

Data	Unit
Temperature	Kelvin
Precipitation	Qualitative
Humidity	Percent
Wind	MPH

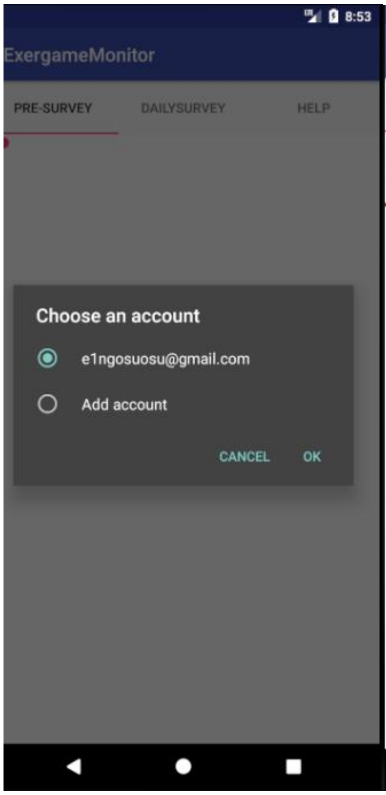
Figure 4.1.1: OpenWeatherMap Data Used

As far as scalability with OpenWeatherMap is concerned, the company limits calls made per minute. The maximum number of free calls per minute is 60. Because of the scale of this project, the application itself made these calls directly to OpenWeatherMap at the start and end of each session. If the scale of the project had been even larger, including thousands of people, it would have been necessary to create a web server to act as a middle man and cache these calls to eliminate extraneous calls.

Using the Android ambient temperature sensor was initially considered, but ultimately rejected, because it had fewer features. Via the Android ambient temperature sensor, a listener can be created to record the temperature whenever it changes. While it lacks many features, the upside to this system was that continuous updates are received as temperatures change and it does not rely on cellular data.

The activity was created in a TabView. The first view showed the pre-questionnaire, and disappeared after the participant responded. the second provided daily EEQ survey, and the third presented an information page. The page showing the pre-questionnaire EEQ were implemented using a WebView. The view simply directed the user to a Qualtrics page showing the study, however, any website with the questionnaire could have been used. Originally it was decided to have a configuration page, but it took up screen page and during pilot studies it was found people didn’t utilize it to change the time their phone uploaded.

To display the proper views, shared preferences was utilized. Shared preferences are accessible key-values stored in a File by android for each app. The preference was initially set to false, indicating first launch. When set to false, only the final tab displayed, but the others were set up as well. After being started, the flag set to true and, on subsequent launches, it was hidden.

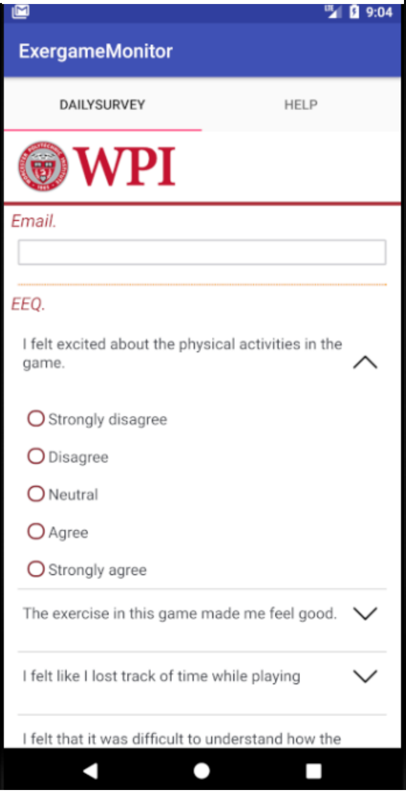


Choose an account

● e1ngosuosu@gmail.com

○ Add account

Choosing Google Fit Account



Email.

EEQ.

I felt excited about the physical activities in the game.

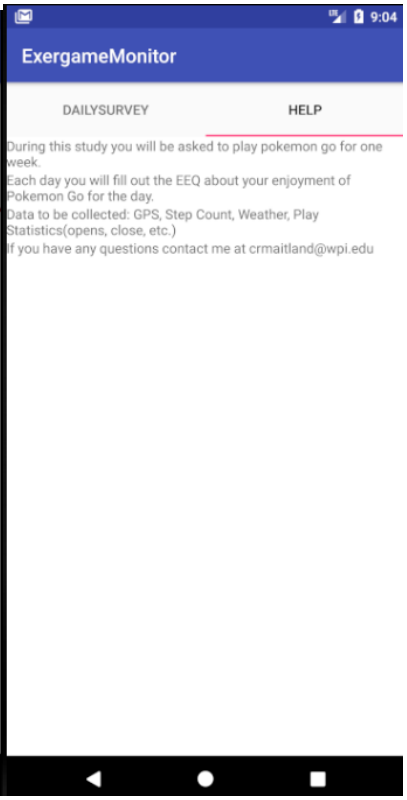
○ Strongly disagree
○ Disagree
○ Neutral
○ Agree
○ Strongly agree

The exercise in this game made me feel good.

I felt like I lost track of time while playing

I felt that it was difficult to understand how the

EEQ View



During this study you will be asked to play pokemon go for one week.
Each day you will fill out the EEQ about your enjoyment of Pokemon Go for the day.
Data to be collected: GPS, Step Count, Weather, Play Statistics(opens, close, etc.)
If you have any questions contact me at crmaitland@wpi.edu

Help View

The uploading of files was done via Dropbox. For this process, Dropbox API v2 was used. Other builds were available through Dropbox's Github at https://github.com/dropbox/dropbox-sdk-java. It is worth noting that the actual Dropbox website features Dropbox API v1, which has been deprecated as of the timing of this report.

Dropbox was also used to obtain participant ID numbers. If a subfolder within the ExergameMonitor remote folder did not exist, the participant was given the ID 1. After this, subsequent participants were given incremental names based on existing folders.

Although Dropbox did not get its own service, it was created as an AsyncTask. AsyncTasks are an android class for starting threads that aren't intended to run for extended periods of time. See A service ran and, at 11:59:59 P.M. or when the user told it to via the info tab view, executed this task. The task then proceeded to upload all files in the directory to the Dropbox folder, while not having to wait for each individual file one by one.

All file writing was handled by the individual services. The services had access to a csv write class that implemented opencsv. This is a csv library for Java that can be downloaded from Sourceforge. Using opencsv, rather than having to worry about how to format data and File IO, you simply tell it which file name you want and the datatype and values.

The only service which did not do this was the weather service. OpenWeatherMap returned all data as a json string and these strings were simply saved as is with a timestamp. They were later processed and turned into csv formats by scripts on a computer.

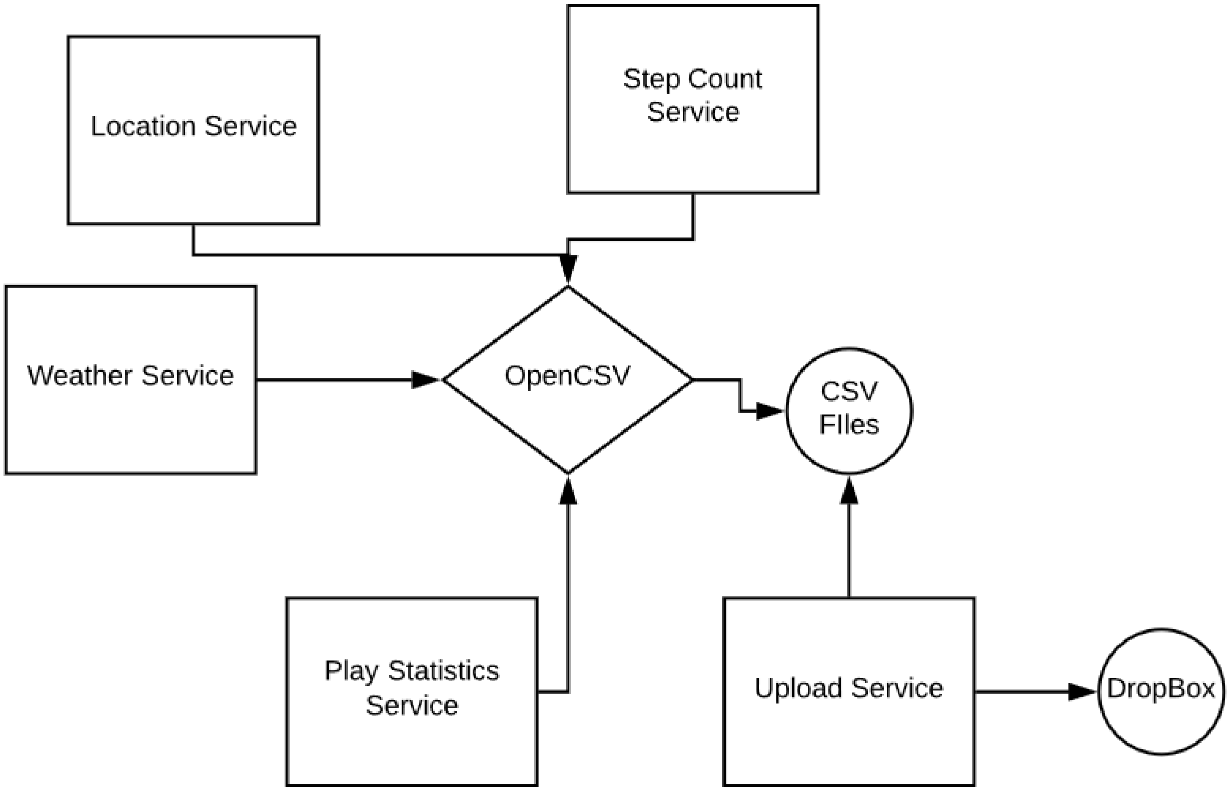


Figure 4.1.1: Service Interaction Diagram

4.2 Python Script to Extract Features

All the data for the process of shallow learning was written and uploaded to the remote Dropbox location via the implemented ExergameMonitor application. The script written in Python 2.7.1 for the second portion of the project transformed the data into usable features and formats.

The first portion of the script collected the data from Dropbox. This was done via the Dropbox API. This could either be installed to Python via pip with the command “pip install dropbox,” or could be built from source on Dropbox’s Python SDK page located at https://github.com/dropbox/dropbox-sdk-python. Another potential method that was not used, but would have also worked, was the Dropbox REST API. The script would have cloned all of the files in the remote ExergameMonitor folder to a directory of the same name nested in the directory of the script. If the script was run multiple times, it would have cleaned the folder of all content and rewritten it from scratch.

One process that could be examined further in the future would be to check whether or not the file had been updated within the remote folder since the last update. This is possible with Dropbox’s getmetadata functionality; however, this study did not examine this. If this were found to have worked properly, the process could simply check which files had been updated or created since the last local update, and only update these.

After the data was backed up on the local directory, the next step taken by the script was to decrypt it. As the files were encrypted in AES, PyCrypto, the Python Cryptography Toolkit was used. Although this package has been dropped and hasn’t been updated since 2014, it features multiple cryptography functions such as SHA256, AES, DES, and RSA. Using this package and the key used in the encryption, the data was decrypted. Installation and basic usage information can be found on PyCrypto’s Github page at https://github.com/dlitz/pycrypto.

Once the data was decrypted, feature extraction was the next step. At this point, feature extraction is simple algebra to extract any desired features using Python addition. Variables such as how much time the user plays each day were figured out by subtracting start times from end times and then summing all of these up throughout the day. This type of methodology holds true for all features.

Example pseudocode of the script is:

```
For each User:
    For Each Feature:
        DecryptFile();
        ExtractFeatureFromData();
        WriteFeatureToUserCSVFile();
For each UserCSVFile():
    AddEachRowToMasterCSVFile();
```

In order to find distances covered, the Haversine formula is used. The haversine formula is used to calculate the distance between two points on a sphere.

$$\operatorname{hav}\left(\frac{d}{r}\right) = \operatorname{hav}(\varphi_2 - \varphi_1) + \cos(\varphi_1)\cos(\varphi_2)\operatorname{hav}(\lambda_2 - \lambda_1)$$

$$\operatorname{hav}(\theta) = \sin^2\left(\frac{\theta}{2}\right) = \frac{1 - \cos(\theta)}{2}$$

Where φ is latitude, λ is longitude, r is radius, and d is distance between the points.

This tries to accommodate the spherical nature of the earth, since the distance from latitude A to latitude B is not necessarily equal to the distance from latitude C to latitude D, even if A-B equals C-D. Here is an example of the implementation of the Haversine function:

```
def haversine(lon1, lat1, lon2, lat2):
   # convert decimal degrees to radians
```

```
    lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2, lat2])
    # haversine formula
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
    c = 2 * asin(sqrt(a))
    # Radius of earth in kilometers is 6371
    km = 6371* c
    return km
```

The purpose of using the haversine function, as opposed to the standard Euclidean distance formula, is to consider the difference in distance between points on the globe. Distance between longitudes and latitudes are not uniform throughout the globe and this methodology accommodates that.

The final step of the script was the compilation of all the features into several sets of csv files. The first set was a list of features per day. Each row corresponded to a day, and each column to a feature. Each file was named participant#_featuresummary.csv. Each row from every participant's file was then added to a master_featuresummary.csv file. This contained all the rows from all of the individual participant files, including an extra column containing the participant's ID number. Like the data files, these feature files were deleted and recreated on recurrent runs of the script.

4.3 Feature Derivations

Note: For all the following equations, x is a data point, i is its index in the dataset, Φ is latitude, and λ is longitude.

- **Daily Step Count**

$$\mathrm{s} = \{[\sum \Delta \mathrm{x_i}] \mid \mathrm{i\ is\ today}\}$$

The sum of changes in step count given that the data is from today.

- **Playing Step Count**

$$\mathrm{s} = \{[\sum \Delta \mathrm{x_i}] \mid \mathrm{i\ is\ today\ and\ x.\,time\ is\ playing}\}$$

The sum of changes in step count given that the data is from today and the user
is playing during x_i.

- **Distance Traveled**

$$\mathrm{d} = \sum \mathrm{haversine}(\mathrm{x_i}, \mathrm{x_{i+1}}) | \mathrm{playing\ for\ x_i}\ \mathit{and}\ \mathrm{x_{i+1}}$$

The sum of the haversine between all points, and their next point given the user is playing Pokémon GO at the time.

- **Maximum Distance from Home**

$$\mathrm{d} = \mathrm{MAX}\big(\mathrm{haversine}(\mathrm{x}_{home}, \mathrm{x}_{i})\big) | \mathrm{\ playing\ for\ x_i}$$

The maximum haversine between data point x and home, given the user is playing while data point x_i occurs.

- **Temperature**

$$\mathrm{t} = \frac{\sum \mathrm{x_i}}{count(i)}$$

The average of temperature data points x_i.

- **Humidity**

$$h = \frac{\sum x_i}{count(i)}$$

The average of humidity data points x.

- **Wind**

$$w = \frac{\sum x_i}{count(i)}$$

The average of wind data points x.

- **Precipitation**

$$p = x_{\max(i)}$$

The last precipitation points x_i where the user is playing.

- **Open Count**

$$0 = \text{count}(x) | \, x.status = open$$

The count of data points where status = open.

- **Earliest Open**

$$0 = \min(\text{x.time}) | \, x.status = open$$

The min of data points where status = open.

- **Latest Close**

$$0 = \max(\text{x.time}) | \, x.status = close$$

The max of data points where status = close.

- **Average Lag Time**

$$t = \frac{\sum x_i - x_{i-1}}{\mathrm{count}(x)}\Bigg| x.\mathrm{status} = \mathrm{open}$$

The sum of start-end times divided by the count of status changes given that the status changed to open.

4.4 Software/API Versions

There were several changes made to API's throughout the app development. Early on, Dropbox v1 was used, but as the project progressed, Dropbox deprecated v1, and the app was migrated to v2. Table 4.3.1 has the API and Software versions utilized.

MATLAB	2017a	https://www.mathworks.com/company/newsroom/mathworks-announces-release-2017a-of-the-matlab-and-simulink-pro.html
Plot.ly	D.O.A: Dec-Feb 2018	Plot.ly
Dropbox API	v2	https://www.dropbox.com/developers/documentation/http/documentation
OpenWeatherMap	Free, DOA: November-Feb 2018	https://openweathermap.org/api
Cohen's Kappa	1.3	https://www.mathworks.com/matlabcentral/fileexchange/15365-cohen-s-kappa

Figure 4.4.1: Final API Versions

4.5 Patch Cycle

Several Patches occurred throughout the apps development cycle. Early on, bugs were detected. Things such as excessive file io were reduced, and SQLite was implemented for data storage until upload time to reduce the need for multiple CSV files containing similar data on the user’s phone. Even things such as how the users played the game forced us to change certain definitions in the game. Figure 4.4.1 is a table of the various changes made throughout the development process.

.1	Beta Versions (Just testing data collection)
1.0	All Data Collection implemented
1.1	Reduced location polling frequency to reduce write frequency
1.2	Implemented Local Data Storage in SQLite, to reduce excess csv generation.
1.3	Add the python script functionality at upload time for app to reduce need for large python script
1.4 (Final)	Redefined “Playing the game” as actively in the app

Figure: 4.5.1: Patch Cycle

5 Results

5.1 Demographics

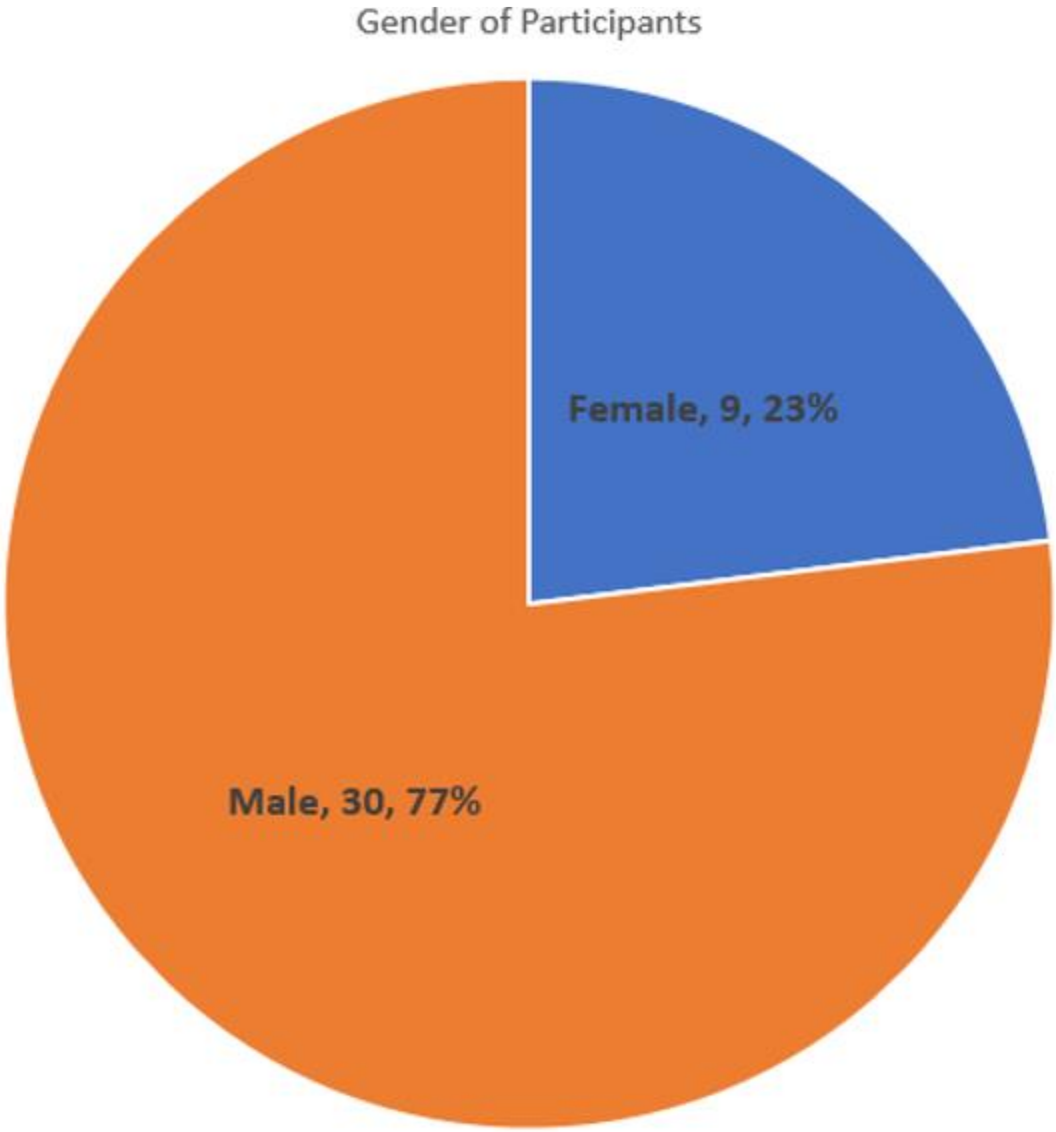


Figure 5.1.1: Gender Distribution of Participants

In figure 5.1.1, we can see the distribution of gender in the study. Of the 39 subjects who participated in this study, 9 were female and the remaining 30 were male.

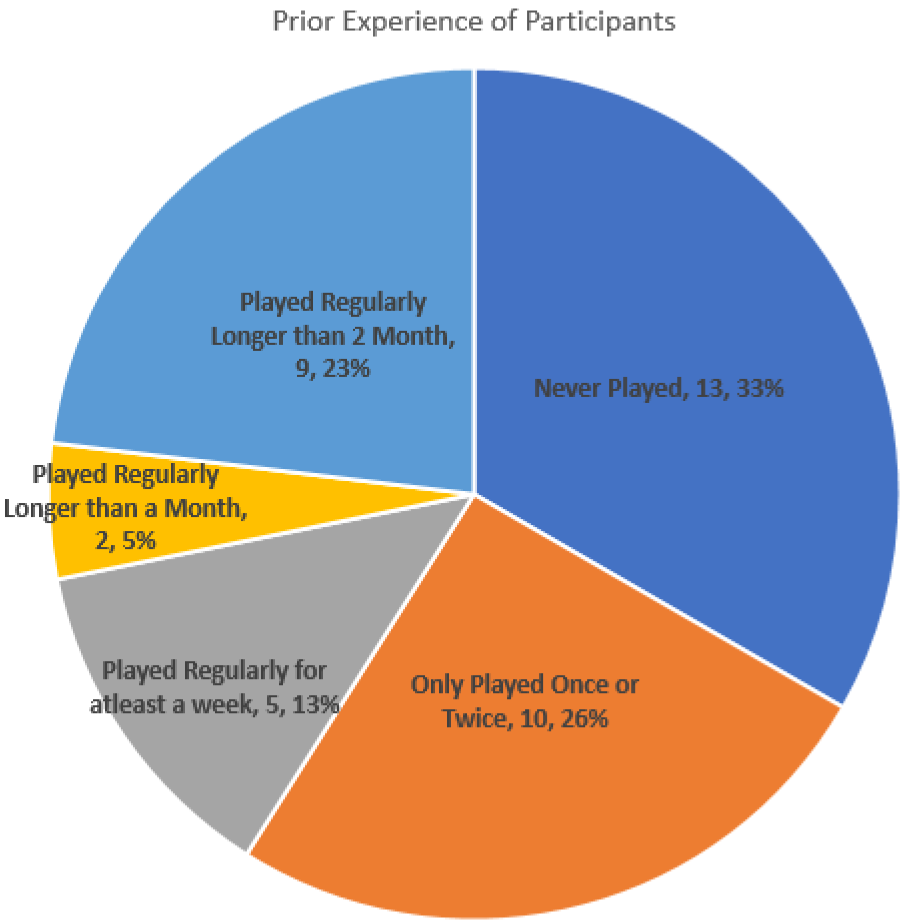


Figure 5.1.2: Prior Experience Distribution of Participants

Figure 5.1.12 shows the distribution of Exergame Experience. For this study, a majority (59%) of the participants had either never played the game before or had only played it once or twice. Included in this group, 13, or 33%, reported that they had never played the game and 10, or 26%, responded that they had played in the past, but only intermittently when they just tried it or played with a friend. 5, or 13% of participants responded that they had played the game for at least a week, while the smallest category was made up of 2 people (5%) who had played regularly for longer than a month. The largest category for regular players was made up of 9 players, representing 23% of the total group who reported playing for regularly for longer than 2 months.

Only broad conclusions can be drawn based on the demographic data. First, most participants were not regular players of the game. Second, of the 41% of players who reported they were regular players, the most (23%) reported being a regular player for more than 2 months.

5.2 EEQ Distribution

In theory, the possible range of EEQ scores is from 20 to 100. In the data we collected, the observed minimum was 45, with a maximum of 90. The observed mean demonstrated by our sample was around 71.

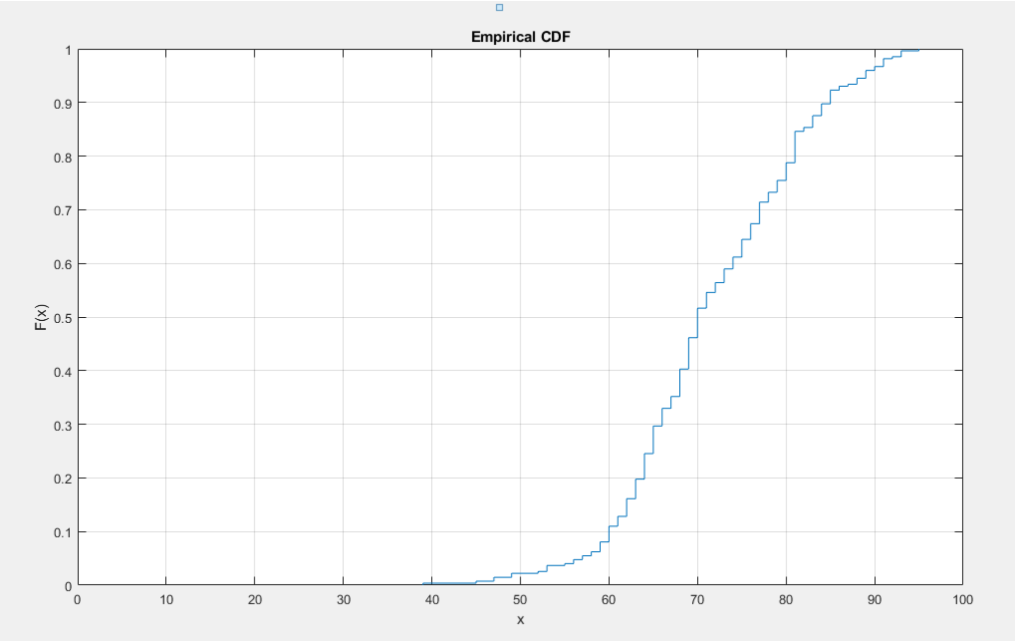


Figure 5.2.1: EEQ Scores CDF Plot

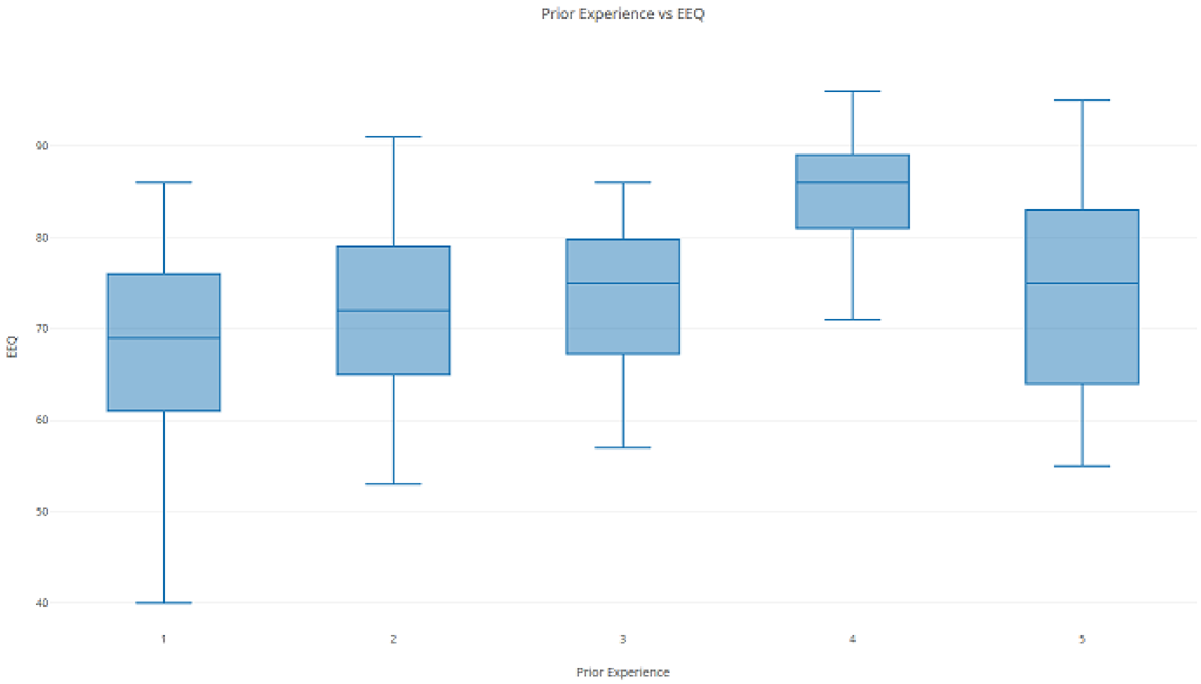


Figure 5.2.2: EEQ vs. Prior Experience

It's interesting to look at the relationship between the prior experience of participants and how they answer the EEQ. The mean EEQ of those who responded they had played for at least a month was 77, the highest EEQ. This makes sense, since a higher EEQ signifies enjoyment of the game and it makes sense that the more a player enjoys the game, the more likely he or she would play it regularly. It is also interesting to compare the medians for each category of regularity of play. In figure 5.2.1 we can see that as the experience of play increases, so too did the median EEQ. The only exception was in the jump to over 2 months played.

5.3 Gameplay Statistics

The first section of features looked at to see how users were playing Pokémon Go was step count. In our data, the mean daily step count was 5,559.5. The data showed two peaks: participants who were less active and players who were very active. The low activity users tended to hover below the 5,000-step mark, sometimes going over it. On the other hand, the high activity users achieved a little over 7,000 steps a day. Very infrequently the data showed players exceeding the 10,000 steps a day that are commonly recommended for a healthy lifestyle.

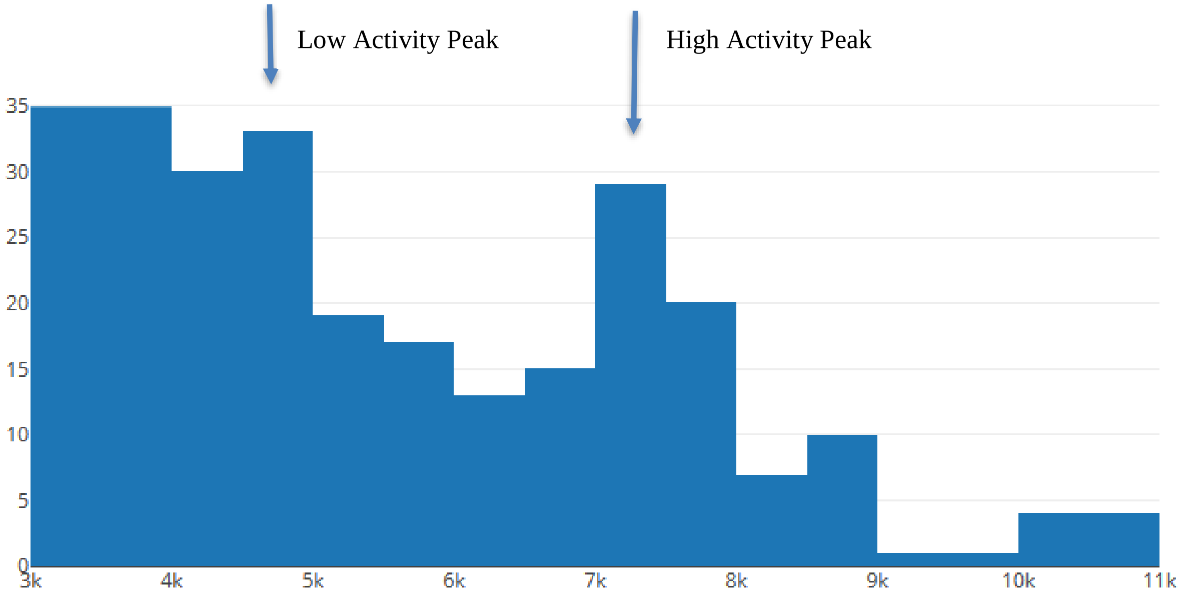


Figure 5.3.1: Daily Total Step Count

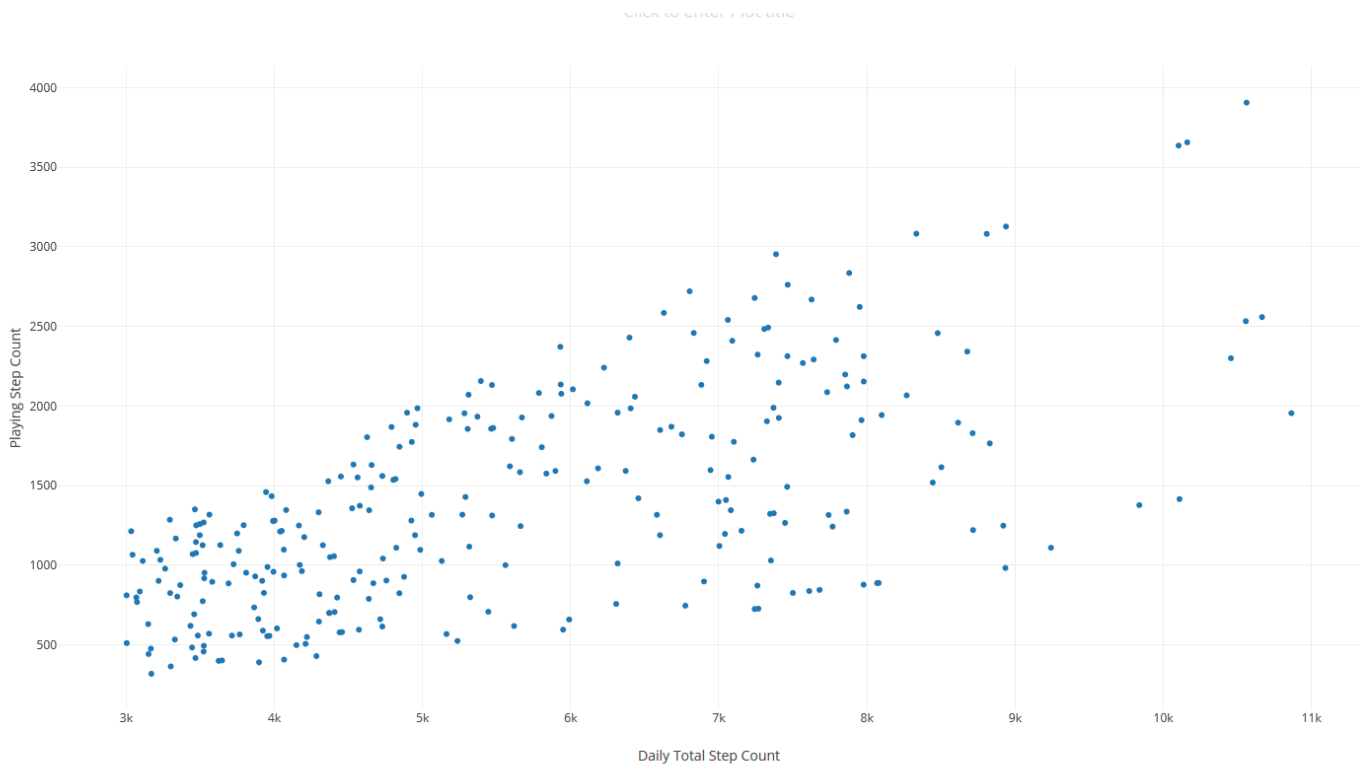


Figure 5.3.2: Daily Total Step Count vs Playing Step Count

It is interesting to note that the daily total steps is closely mapped to the steps you take while playing the game, showing a correlation of .6459. One initial impression I had was that there would be a cap on daily step count. I thought if you were more active during the game, this simply meant you would be less active at other times in the day. This tended not to be the case though with both increasing together. As you increase the playing steps, it simply adds to the total steps without a clear cap.

Precipitation was one feature that didn't present much in this study. This is largely because it didn't rain. Only 12 out of the 274 feature sets had it have some form of precipitation, and this was largely just light rain. To get quantitative data for this I simply used the values returned by openweathermap as a metric. Light rain was given as the value of 1, rainfall 2, and heavy rain 3.

One thing noted during the study was the difference between how people traveled while playing the game. When looking at all the distance traveled data we see a large cluster of low data, with a skew to the right of those whose distance traveled is drastically larger. When asking participants who had larger skews about this, they informed me that they utilized Pokémon Go Plus. Pokémon GO Plus is a wearable that allows the user to be notified when they’re close to a Pokémon and catch Pokémon without having to go into their phone. Using this, they could play the game while driving to work and other locations because they didn’t have to look at their phone. This resulted in a much different play style. Although people without this would drive some distance, it was a drive and then once they got their they started actively playing the game. This had a similar effect on time played, where the median was 99.5 minutes and the mean was 220 minutes, simply because they didn’t exit the app while they did their daily routines. This resulted in the decision to re-define what “playing” Pokémon Go constitutes. Before the idea was that if the app was open, the user might exit out of it but still be playing games. After collecting data, it is updated to be if a player is in the Pokémon Go app, or the last thing their phone was on sleep was in the app.

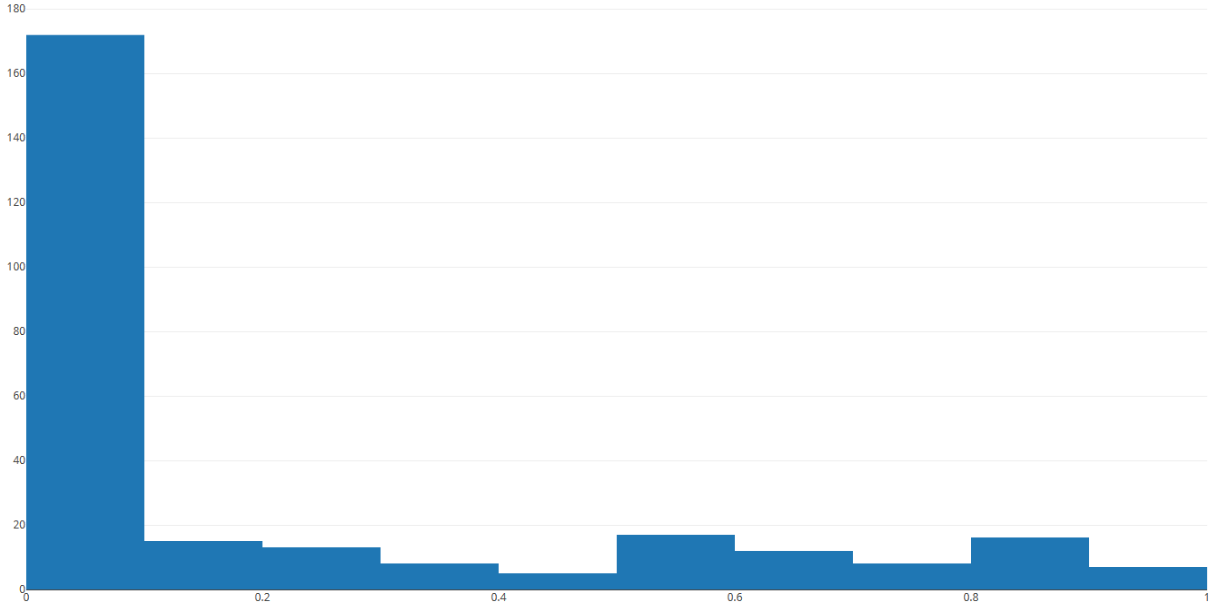


Figure 5.3.3: Playing Step Count

Feature	Mean	Median	Standard Deviation
EEQ	72.0476	72	10.3042
Daily Total Steps	5595.2124	5181	1881.3048
Playing Steps	1389.2527	1265	679.7110
Distance Traveled	0.2064	0.016	0.2978
Distance Traveled From Home	0.3242	0.31	0.1129
Temperature	46.2868	41	12.2171
Precipitation	0.0732	0	0.3757
Wind	7.7271	8.1	5.8683
Humidity	55.4835	60	12.0213

Times App Opened	13.8095	14	3.0503
Time Played	72.9871	73.14	12.2478
Average Session Length	19.6520	20	2.9895
Earliest Open	10.2733	10.2982	0.9496
Latest Close	19.2417	19.2556	1.1461
Average Lag Time	178.4018	194.0182	27.8192

Figure 5.3.4: Feature Statistics

Figure 5.3.4 allows us to see the distribution of all the statistics. This allows us to see the interesting results of some the results of our data. Using this, we can see how it's hard to use a feature such as precipitation due to its mean being a value of 0. This means that at least 50% of the data points indicate no precipitation. This is further illustrated by figure 5.3.5.

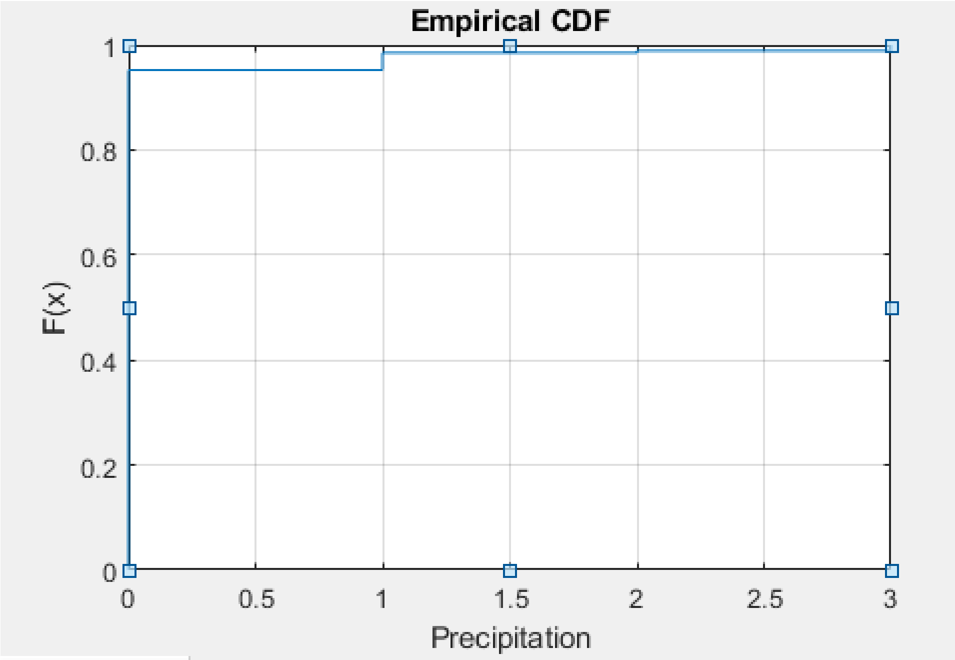


Figure 5.3.5: Precipitations CDF

5.4 Feature Selection

Ron Kohavi defines features interchangeably with attributes. He says an attribute is “A quantity describing an instance. An attribute has a domain defined by the attribute type, which denotes the values that can be taken by an attribute” [27]. As a hypothetical, if a model was to be trained to determine whether a day was going to have rain or not, you could use the time of year and what the humidity is as features.

When conducting a study, some data collected can be found to be irrelevant. These features simply add noise to any models produced, resulting in the possibility of less accurate results being acquired. Because of this, it is necessary to prune out the features that are found to be statistically insignificant.

During this study, a total of 15 features were collected. To determine which of these features are relevant, we utilized the P value collected from using the t-test. The cap on those values which are believed to be statistically significant is .05. Features with values lower than this are considered statistically significant, which means that they can predict the EEQ, while those with higher values than this are not and, thus, cannot be used to predict the EEQ.

Feature	P Value	Correlation Coefficient
Daily Total Steps	<.00001	-.6153
Playing Steps	<.00001	.5512
Distance Traveled	<.00001	.5654
Distance Traveled From Home	<.00001	.4451
Prior Experience	<.00001	.7012
Time Played	<.00001	.2318

Times App Opened	.02830	.6889.
Average Session Length	.04210	.2481
Latest Close	.17820	-.4210
Average Lag Time	.3162	-.2132
Earliest Open	.3817	.0041
Precipitation	.20173	-.1386
Temperature	.49377	.1937
Wind	.88421	-.0701
Humidity	.96650	.0083

Table 5.3.1: Feature Correlation to Exergame Enjoyment Score

Those features that are found to not be statistically significant are not included as classification features.

It is interesting to look at which features ended up being statistically significant. Features which end up directly correlating to how the game functions were found to be statistically significant. For example, if you look at something such as how much distance is traveled by the players, this can be mapped to how much a player would have to move to get to a gym or poke stop. Daily total steps are similar to steps while playing because in general the steps people take while playing map to daily total steps in addition. Normally playing steps added to total steps, they did not subtract from it.

To the contrast, some other features about how they played it didn’t really show any relationship. For example, some people chose to play it on and off throughout the day, others chose to just do one longer session after work or classes. This is mostly how it could be fit into their schedule as opposed to how much they enjoyed the activity.

5.5 Data Processing and Classifiers

5.5.1 Data Processing

In a study such as this, we are attempting to predict how highly a user rates their enjoyment playing the game on any given day. These results are represented by the EEQ value, a number ranging between 20 and 100. In order to classify this data, you must break it up into bins that categorize these scores.

For example, one metric would be to use the median value as a bin. EEQ's below the median score would represent a day that the player did not enjoy the game and higher EEQ's would signify a greater level of enjoyment if the classifier was perfectly accurate.

When running analysis, bins of various sizes were chosen to be used as our classes or groupings. The initial two bins divided the data up along the mean and median of the EEQ scores. Scores below the median EEQ score of 72 would be put into bin 'A', and those above would be put into bin 'B.'

Various bin sizes were tried in the utilization of classifiers. One way we chose to break the data up was along the mean and median. Here a score of below the mean would be put into bin 'A' and the scores above would be in bin 'B'. Bins of set sizes were also used. For example, a bin of size 20 would have the EEQ scores below 20 in bin 'A', between 20 and 40 would be in bin 'B', etc.

Utilizing the features deemed statistically significant, we aim to create a model that can predict which bin the subject should go into as accurately as possible. If bin 'D' is set at the EEQ score range of 60-80, and a person's survey responses determine an EEQ of 67, they should be classified as being in bin 'D' by our model. Figure 5.5.5.1 has the final bins used.

Bins Used	
Mean	0-71, 71-100
Median	0-72, 72-100
Bin 20	0-20, 20-40, 40-60, etc.
Bin 10	0-10, 10-20, 30-40, etc.
Bin 5	0-5, 5-10, 10-15, 15-20, etc.

Figure 5.5.5.1: Bins Used

5.5.2 Classifiers Utilized

For us to utilize MATLAB and run the classifier for results, there were two primary parts used: the classification learner and scripts used to evaluate things such as Cohen's Kappa, a measure of how well our model agrees with the true answer.

The classification learner comes with numerous built-in classifiers that we can use in order to train a model. These include methods such as decision trees, discriminants, and support vector machines. In addition, it also provides variations on different classification models, such as linear, quadratic, and cubic svm's. Utilizing these classifiers, we can build a model which attempts to predict the correct class based on their features, as well as to see how the model performed in a confusion matrix.

The second part of data analysis came when determining Cohen's kappa, as well as other measures of how well the classifier performed. These came from a MATLAB function created by Giuseppe Cardillo[28]. We could run this function on the confusion matrix generated by the classifier and view how the classifier performed. The data produced by this function is in figure 5.4.2.1.

Returned Value	
Observed Agreement	Proportion of classifications agreed upon
Random Agreement	How likely the two are to randomly agree.
Agreement due to True Concordance	How much you agree that’s not random.
Residual not Random Agreement	1-Random agreement
Cohen’s Kappa	One of the various kappa statistics, showing the agreement of two classifications.
Kappa Error	The standard error of kappa
Kappa C.I.	The confidence interval for kappa
Maximum Possible Kappa	Upper bound of kappas confidence interval
K observed as proportion of maximum possible	What the observed is compared to the maximum on it’s confidence interval
Variance	Variance found for kappa
Accept Null Hypothesis	Whether or not the agreement is accidental

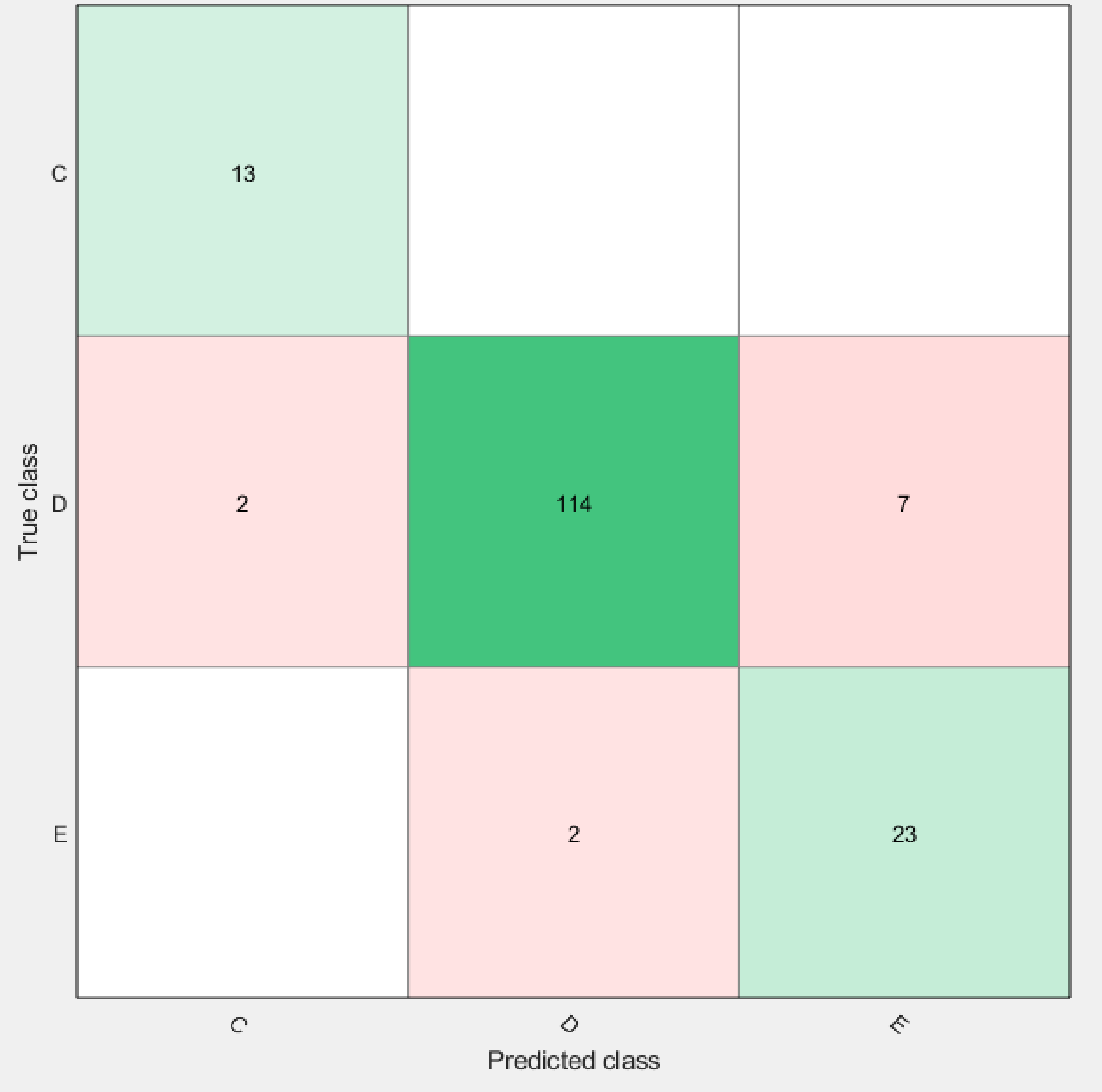
Figure 5.4.2.1: Giuseppe Cardillo’s Matlab Kappa Function Output

5.5.3 Confusion Matrices and Cohen’s Kappa

A confusion matrix is a table that shows the visualization of how well a classifier has performed on a set of data. It summarizes the performance of a classification algorithm. Each column in the table corresponds to a value predicted by the model. Each row in the table corresponds to what the true value should be. Utilizing both of these, we can see how well the model correctly predicted the classes for the data set given.

Similarly, we can use this matrix to calculate Cohen’s kappa, a quantitative measure of how well two models compare to each other. Cohen’s kappa measures the agreement between variables. It can be used to compare how well different raters classify subjects into groups. Jacob Cohen was troubled using percent agreement, because it did not account for chance agreement. So, in 1960 he developed the Cohen’s kappa to consider the possibility that raters guess on some variables due to uncertainty.

Figure 5.4.3.1 illustrates an example of a confusion matrix. Here, we can see that the model predicted 2 values as D, when they should be C, 2 as E that should be D, and 7 as D that should be E. In this, class C indicates a score of 40-60, class D a score of 60-80, and class E a score of 80-100.



True class \ Predicted class	C	D	E
C	13		
D	2	114	7
E		2	23

Figure 5.4.3.1: Sample Confusion Matrix

After finding the confusion matrix, it is possible to calculate Cohen’s kappa. The purpose of the kappa statistic is to see how closely two different observers of a set of data will classify that data. In the case of our classifiers, we are looking to see how closely our model agrees with the true value. A kappa of 1 would indicate that they agree perfectly, while a kappa of 0 would imply they don’t agree. A kappa of -1 would indicate the two were in complete disagreement.

5.5.4 Classification Results

Median Bin		
Classification Model	Correctly Classified	Kappa Statistic
Fine Tree	61.2%	0.2240
Course Tree	63.4%	0.2605
Linear Discriminant	50.9%	0.0120
Linear SVM	52.0%	0.0316
Quadratic SVM	53.5%	0.0656.
Cubic SVM	57.5%	0.1483
Fine Gaussian SVM	55.7%	0.0931
Fine KNN	49.8%	-0.0062
Course KNN	52.0%	0.0234
Boosted Trees	61.9%	0.2360
Bagged Trees	62.3%	0.2449

Mean Bin		
Classification Model	Correctly Classified	Kappa Statistic
Fine Tree	63.7%	0.2646
Course Tree	65.9%	0.3115
Linear Discriminant	57.5%	0.1183
Linear SVM	54.2%	0.0523
Quadratic SVM	59.7%	0.1767
Cubic SVM	60.8%	0.2119
Fine Gaussian SVM	55.7%	0.0431
Fine KNN	58.6%	0.1374

Course KNN	54.6%	-0.0181
Boosted Trees	64.8%	0.2802
Bagged Trees	66.3%	0.3149

Bin size 20

Classification Model	Correctly Classified	Kappa Statistic
Fine Tree	65.2%	0.1899
Course Tree	71.8%	0.2235
Linear Discriminant	72.5%	.1494
Linear SVM	72.9%	0
Quadratic SVM	71.1%	.1648
Cubic SVM	58.6%	.0321
Fine Gaussian SVM	72.5%	-.0066
Fine KNN	57.5%	-.0584
Course KNN	72.9%	0
Boosted Trees	73.6%	.2789
Bagged Trees	74.4%	.2602

Bin size 10

Classification Model	Correctly Classified	Kappa Statistic
Fine Tree	41.8%	0.1378
Course Tree	48.0%	0.2136
Linear Discriminant	FAILED%	FAILED
Linear SVM	37.7%	0.0527
Quadratic SVM	41.8%	0.1186
Cubic SVM	35.9%	0.0538

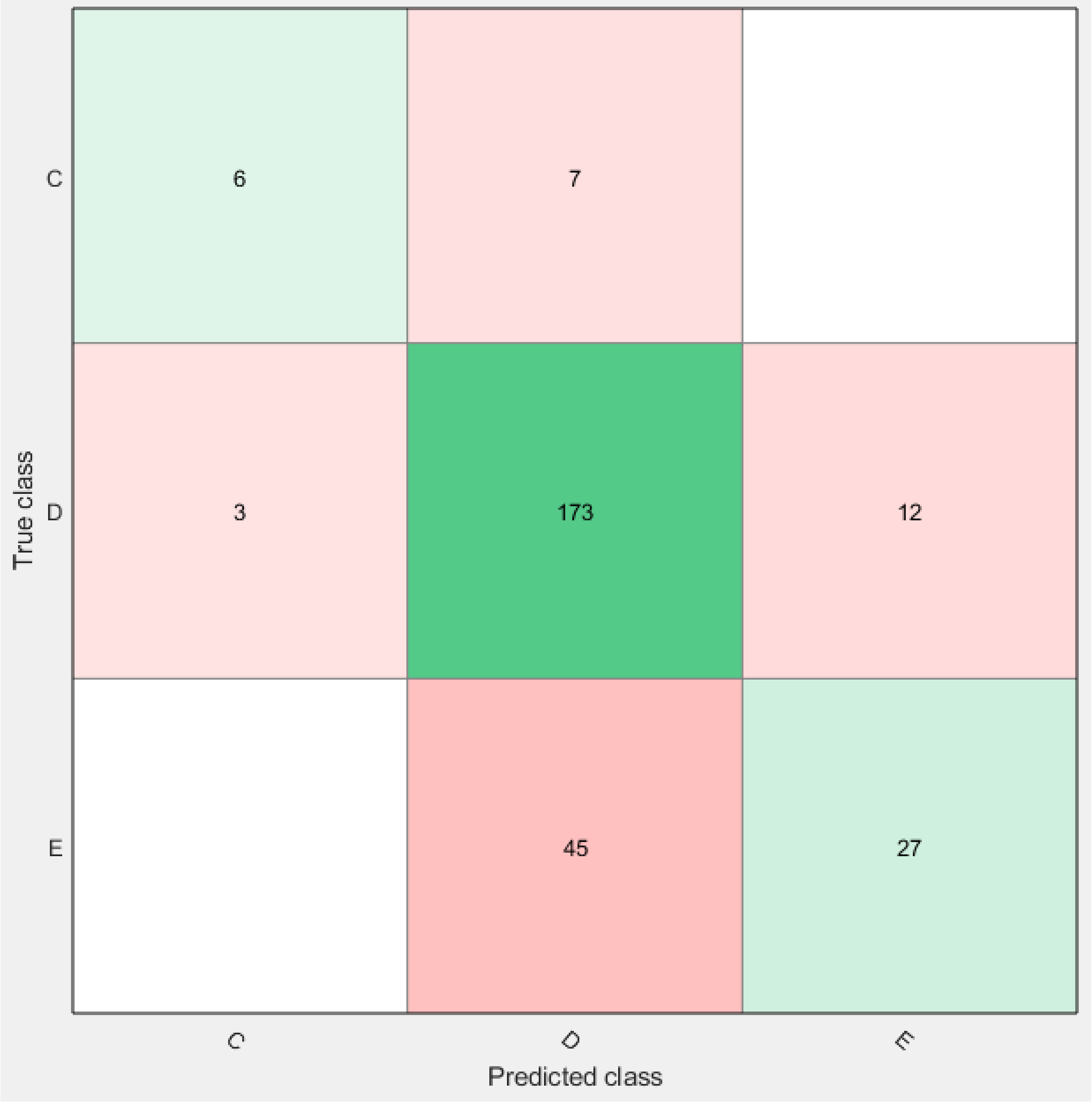
Fine Gaussian SVM	35.9%	0.0686
Fine KNN	34.4%	0.0417
Course KNN	39.6%	0.0452
Boosted Trees	42.9%	0.1473
Bagged Trees	45.8%	0.1872

Based on the results in all bin sizes, besides bin size 10, it becomes evident that Bagged Trees is a solid baseline for us to use as a model for classifying the users' enjoyment of the game. It demonstrated the highest percent of accurate classifications of any of the classifiers for both the mean, median, and bin size of 20. It peaked in accuracy with a bin size of 20, giving it a correct classification rate of 74.4%.

It is interesting to note that feature selection cleans up the models produced in our figures. When looking at the bin size of 20, for example, the prediction accuracy of Bagged Trees drops from 74.4% to 70.0%.

Bin size was also an important decision when looking at relevant data. Smaller bin sizes were attempted however after the bin size of 10, the accuracy became miniscule with values in the .05 to .2 range.

It is also important to note that although the kappa value indicates there is not random agreement between Bagged Trees and the true value, it is only of fair strength. The kappa statistic peaked at .31, falling into the .2 to .4 bracket. Those in this range can be interpreted as a fair agreeance. Ideally though, you would see a value of near .6 or higher, which would indicate a substantial agreement between the two.



True class \ Predicted class	C	D	E
C	6	7	
D	3	173	12
E		45	27

Figure 5.4.4.1: Bin size 20, Bagged Tree Confusion Matrix
C = 40-60, D = 60-80, E = 80-100

6 Conclusion

The goal of the CyPRESS (Cyber Physical Recommender System) project is to identify the variables that affect the enjoyment of people playing exergames, with the goal of helping the retention of people playing exergames. People in the United States do not get enough physical activity and this is causing numerous health problems, such as heart disease and diabetes. Exergames are a way to increase physical activity using mobile technology.

However, statistics tell us that there is a massive issue with the drop off in players of exergames such as *Pokémon GO*. Very often, players stop playing after a period. Core to fixing this issue is determining if a player is likely to stop playing the game by identifying the variables associated with that behavior. Based on this study's results, it is possible to use features collected to attempt to predict where along the chain of retention users are.

6.1 The Model

Based of the results of this study, bagged trees turned out to be the most efficient at classifying the data under multiple scenarios. With bins split around the mean, median, and with bin size of 20, it proved to be the most accurate classifier. Bagged trees peaked with a bin of size 20, where it could accurately predict 74.5% of the bins correctly. Because of this, I believe bagged trees to be the most suitable model to use as a baseline. However, there are some considerations one must make.

When classifying this data, it is important to consider what the bin size is for every classifier. If the bin size is too granular the model will overfit the data. This results in the model accounting too much for noise in the present data set and becoming overly complicated based on current data. Similarly, we want to allow enough bins to break up the data into. This was clearly demonstrated by the results of testing various bin sizes. With the mean and median as the cutoff for bins, the results were less accurate than with a bin of size 20. On the other hand, as the bin size decreased from 20 to 10 to 5, the results of the classifiers became more and more inaccurate. This is useful in determining to what degree a user is enjoying a game. You may break up the EEQ into different levels of enjoyment, but it must be certain that there is no over creation of levels of enjoyment.

6.2 Study Limitations

There were two primary issues I had with this study. These were:

- The lack of duration an individual participated in the study
- The lack of demographics to target the study

One issue with this study is that it takes place over the course of a single week. Looking at data over this short time period does not necessarily allow us to predict player behavior over much longer periods of time. Because the purpose of the study is to impact player retention over the long term, this is certainly a limitation. In past studies, as well as in this study, it was found that most participants played less than a week. Understanding the retention problem requires studying subjects over a longer period. Some subjects may decide to stop playing after a duration of a week and a half, and this study doesn’t account for that.

Similarly, some features which were initially expected to have a correlation, do not vary much within a single week period. For example, the feature of humidity typically does not change much over one week in the winter, but may affect results in the summer.

Another change that would make this study more insightful would be to match up the participants of the study to the actual demographics of those who play *Pokémon GO*. Knowing traits of all *Pokémon GO* players, such as the level of physical activity they get, would be useful to pair up with and set more concise recruitment guidelines. It is important to recruit members who fit the target demographic for the game itself. As an extreme example if we recruit people completely outside of the games target demographics, such as those with agoraphobia who we wouldn't expect to go outside to play the game, it's hard to get data on the player base. They may rate it much lower than the average player would, and although this is good information to know, misses the general trends of users.

6.3 Future Steps

6.3.1 Longer Duration Study

To make the study as accurate as possible, extending the time period to one year would be ideal. A longer period would better capture player's behavior. Currently the study is fast with a participant only participating for a week. Although this is a good tool for creating a methodology, it is a rather limited time to see how participants exhibit changes in behavior. By using a longer-term study, we could see data on events such as when a user quits, and if they do quit what's the likelihood they start playing again, if they do start playing again how is it different, etc.

6.3.2 Collect more data to get a representation of what entails a lower EEQ

It would also be interesting to get more data on people who do not enjoy the game. During this study, the lowest EEQ score given was a 45, while the theoretical lowest on the questionnaire is 20. It would be interesting to accommodate for the data in this dead zone of 20-45, where there is currently no data.

6.3.3 Look at the rate of change of a user's EEQ

Currently this study sought to see if we could implement something to classify the data into various bins. One area it lacks on is seeing how this data changes over time. What sorts of data to look at to see how one user moves from one classification to the next would be invaluable in future studies.

6.3.4 *Pokémon GO* Integration

Something that would be interesting to implement are data features within the app to track *Pokémon GO* accomplishments. In addition to the real-world effects (virtual reality) of a *Pokémon GO* type game, part of what determines how much a player likes an exergame are the things that occur in the game. For example, it would be interesting to note things such as how many Pokémon the players caught, if they played with others or alone, or how often they went to the gym. Similarly, in other games such as *Just Dance,* you could look at how often a player improves his or her high score on a song as a feature.

References

[1]E. Agu, M. Claypool, “Cypress NSF Proposal”, *Worcester Polytechnic Institute*

[2]T. Clarke, T. Norris and J. Schiller, "Early Release of Selected Estimates Based on Data From the 2016 National Health Interview Survey", *National Center for Health Statistics*, 2017.

[3]T. Fakhouri, J. Hughes, V. Burt, M. Song, J. Fulton and C. Ogden, "Physical Activity in U.S. Youth Aged 12–15 Years, 2012", *Center for Disease Control and Prevention*, 2017. [Online]. Available: https://www.cdc.gov/nchs/data/databriefs/db141.htm#x2013;15%20Years,%202012. [Accessed: 02- Oct-2017].

[4]“ACSM Information on Exergaming”, *American College of Sports Medicine,* https://www.acsm.org/docs/brochures/exergaming.pdf [Accessed: 20-March-2018]

[5] Y. Chao, Y. K. Scherer and C. A. Montgomery, "Effects of Using Nintendo Wii™ Exergames in Older Adults: A Review of the Literature," *Journal of Aging and Health,* vol. 27, *(3),* pp. 379-402, 2015.

[6]C. Neustaedter, A. Tang and T. K. Judge, "Creating scalable location-based games: lessons from Geocaching," *Personal and Ubiquitous Computing,* vol. 17, *(2),* pp. 335-349, 2013.

[7]P. Tassi, "Believe It Or Not, 'Pokémon GO' Has 65 Million Monthly Active Players", *Forbes*, 2017 [Online]. Available: https://www.cdc.gov/nchs/data/databriefs/db141.htm#x2013;15%20Years,%202012. [Accessed: 20- March- 2018].

[8]K. Anderton, "Augmented Reality, The Future, And *Pokemon Go* [Infographic]", *Forbes*, 2016. [Online]. Available: https://www.forbes.com/sites/kevinanderton/2016/11/14/augmented-reality-the-future-and-pokemon-go-infographic/#7f9a730a7e98. [Accessed: 26- Sep- 2017].

[9]A. Farvardin, E. Forehand, "Geocaching Motivations", *Worcester Polytechnic Institute*. 2013

[10]W. Garney, A. Young, K. McLeroy, M. Wendel and E. Schudiske, "A Qualitative Examination of Exergame Motivations in Geocaching", *Games for Health Journal*, vol. 5, no. 1, 2017.

[11]T. Leaver, M. A. Willson and DOAB: Directory of Open Access Books, *Social, Casual and Mobile Games: The Changing Gaming Landscape.* (1st ed.) 2016.

[12]S. Mazor and P. Salmon, "Anecdotes: Magnavox and Intel: An Odyssey and The Early Days of the Arpanet," *IEEE Annals of the History of Computing,* vol. 31, *(3),* pp. 64-67, 2009.

[13]M. D. Finco and R. W. Maass, "The history of exergames: Promotion of exercise and active living through body interaction," in 2014, . DOI: 10.1109/SeGAH.2014.7067100.

[14]A. Ohlheiser, “Atari nearly introduced the world to fitness gaming 30 years ago,” *The Washington Post*, 22-Dec-2014. [Online]. Available: https://www.washingtonpost.com/news/to-your-health/wp/2014/12/22/puffer-the-great-atari-exercise-bike-that-never-was. [Accessed: 12-Oct-2017].

[15] Anonymous "Japanese Wii sales neared 250,000 in August," *The Online Reporter (Rider Research), (555),* pp. 13, 2007.

[16]Anonymous "ValueWalk: *Pokemon GO* Popularity Crushed Even Google's Infrastructure," *Newstex Global Business Blogs,* 2016.

[17]A. Bhattacharya, "Android just hit a record 88% market share of all smartphones", *Quartz*, 2017. [Online]. Available:

[18]D. Allan, "Google’s Andromeda OS has reportedly been canned", *TechRadar*, 2017. [Online]. Available: http://www.techradar.com/news/googles-andromeda-os-has-reportedly-been-canned. [Accessed: 04- Oct- 2017].

[19]“Pixel 2 Tech Specs,” *Google Store*. [Online]. Available: https://store.google.com/product/pixel_2_specs. [Accessed: 12-Oct-2017].

[20]“Open mHealth Shimmer,” *GitHub*, 06-Oct-2017. [Online]. Available: https://github.com/openmhealth/shimmer. [Accessed: 12-Oct-2017].

[21]"What is Machine Learning? A definition", *Expert System*, 2018 [Online]. Available: http://www.expertsystem.com/machine-learning-definition/. [Accessed: 21- March- 2018].

[22]S. Theodoridis, Skillsoft Books ITPro Collection and Safari Books Online, *Machine Learning: A Bayesian and Optimization Perspective.* 2015.

[23]H. Xie, S. Devlin, and D. Kudenko, “Predicting Disengagement in Free-To-Play Games with Highly Biased Data,” *Player Analytics: Papers from the AIIDE Workshop*, pp. 144–147.

https://qz.com/826672/android-goog-just-hit-a-record-88-market-share-of-all-smartphones/. [Accessed: 04- Oct- 2017].

[24]A. Elliot, S. Huang, K. Sposato, D. Wang, “Run For Fun - Measuring Exergame Enjoyment”, *Worcester Polytechnic Institute*, 2017

[25]"Plotly", [Online]. Available: plot.ly. [Accessed: 21- March- 2018].

[26]"Classification Learner", *MathWorks*, [Online]. Available: https://www.mathworks.com/help/stats/classificationlearner-app.html. [Accessed: 21- March- 2018].

[27]R. Kohavi, F. Provost, "Glossary of Terms Special Issue on Applications of Machine Learning and the Knowledge Discovery Process", *Stanford*, 1998 [Online] Available: http://robotics.stanford.edu/~ronnyk/glossary.html [Accesed: 22- March- 2018].

[28]Cardillo Giuseppe, “Cohen’s Kappa version 1.3”, MATLAB File Exchange, https://www.mathworks.com/matlabcentral/fileexchange/15365-cohen-s-kappa?focused=5143939&tab=function [Accessed: 10-Feb-2018]

Appendix

1 Exergame Enjoyment Questionnaire

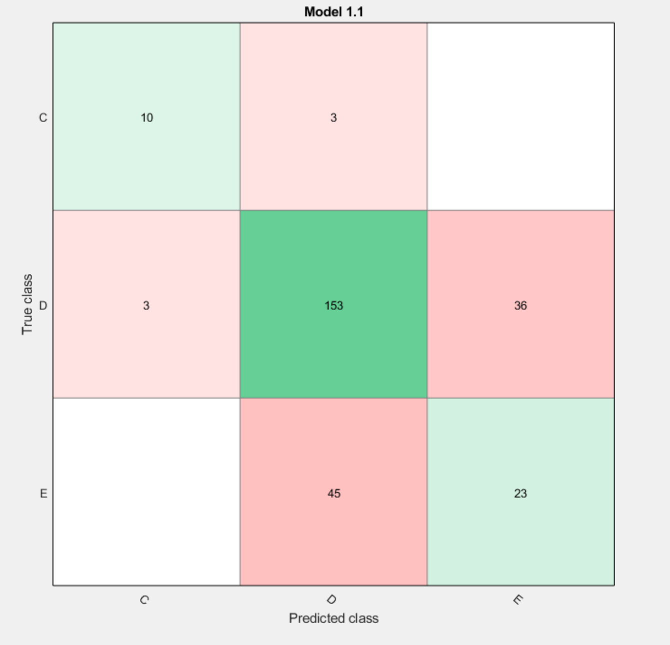
I felt excited about the physical activities in the game.
The exercise in this game made me feel good.
I felt like I lost track of time while playing
I felt that it was difficult to understand how the game works.
I was focused on the game.
I felt that the game would have been more enjoyable without physical activity.
I felt that it was easy to familiarize myself with the game controls.
I felt emotionally attached to the game.
I consider playing the game “exercise”.
I felt that the physical activity was too intense for me.
I did not feel a desire to make progress in the game.
I felt a strong sense of being in the world of the game to the point that I was unaware of my surroundings
I would rather not be exercising, even though the exercise was accompanied by game elements.
I felt that playing the game was beneficial for my physical well-being.
I felt that this game provided an enjoyable challenge.
I felt a sense of accomplishment from playing the game.
I felt that the game reacted quickly to my actions.
.I did not feel like I wanted to keep playing.
I would prefer that this physical activity was not accompanied by game elements.
I felt in control of the game.

2 Pre-Questionnaire

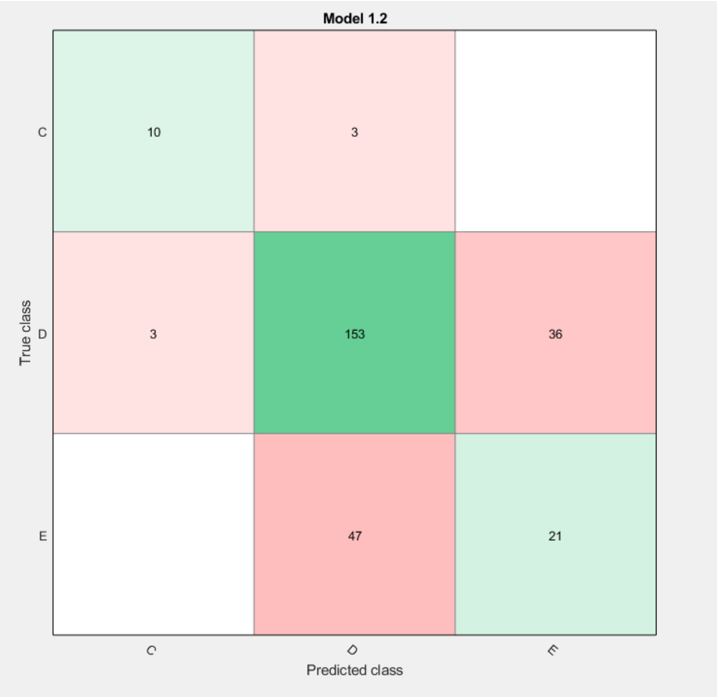
What is your Gender?
What is your Age?
What address would you like to be your home?
How much exercise would you say you get in a typical week?
Have you ever played a video game before?
Have you ever played a game on your phone before?
Do you consider yourself as having played an exercise game before?
Have you played *PokemonGO* before?
If you have, how long did you play?

3 Confusion Matrices

3.1 Bin Size 20

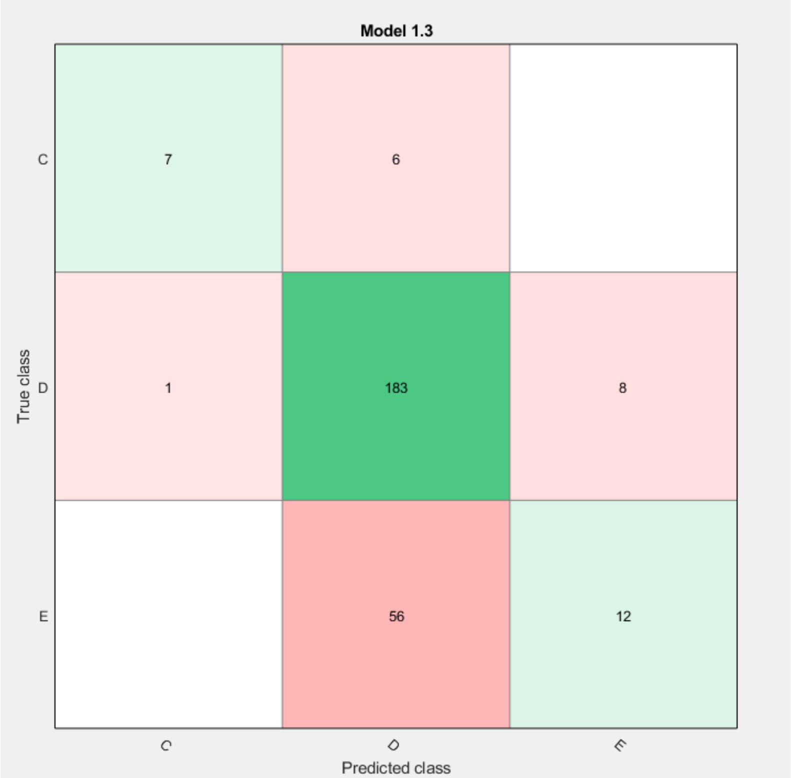


Fine Tree



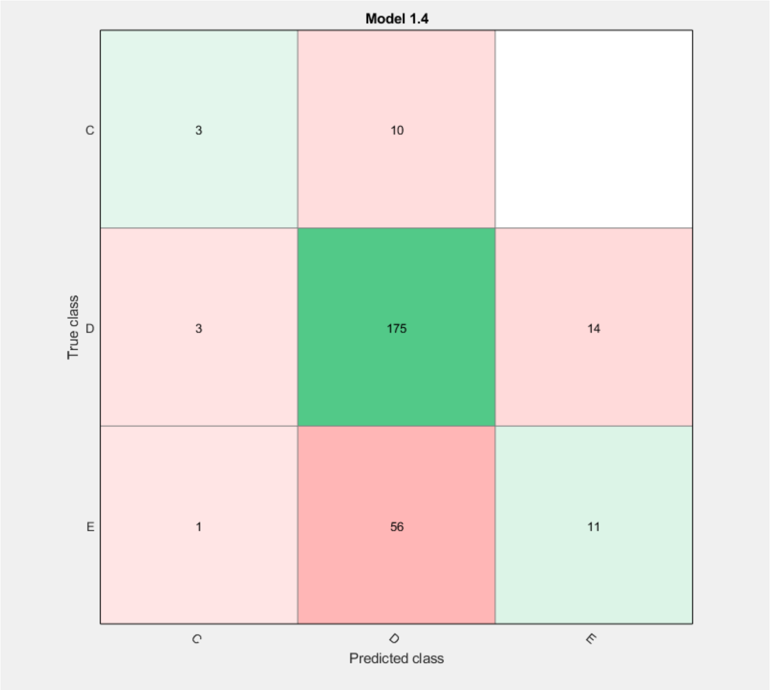
True class \ Predicted class	C	D	E
C	10	3	
D	3	153	36
E		47	21

Medium Tree

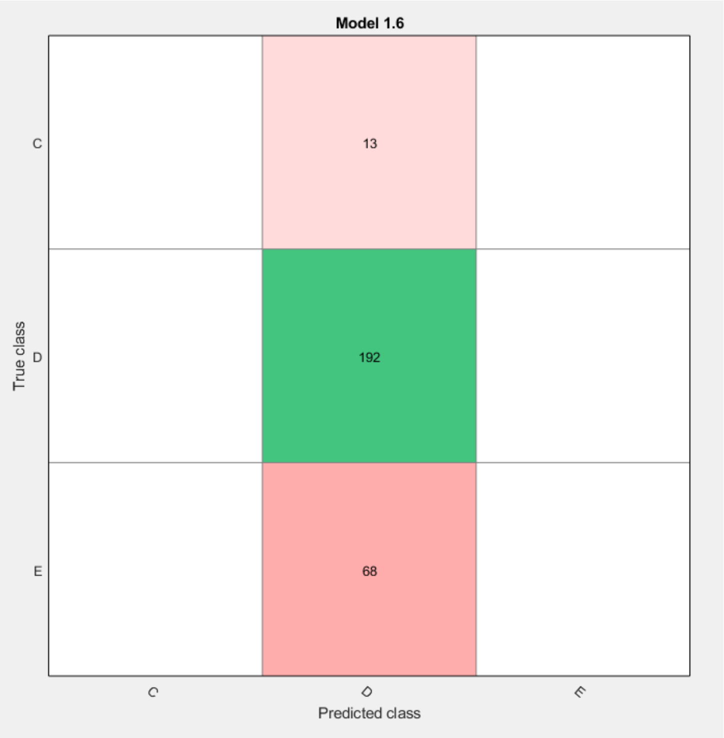


True class \ Predicted class	C	D	E
C	7	6	
D	1	183	8
E		56	12

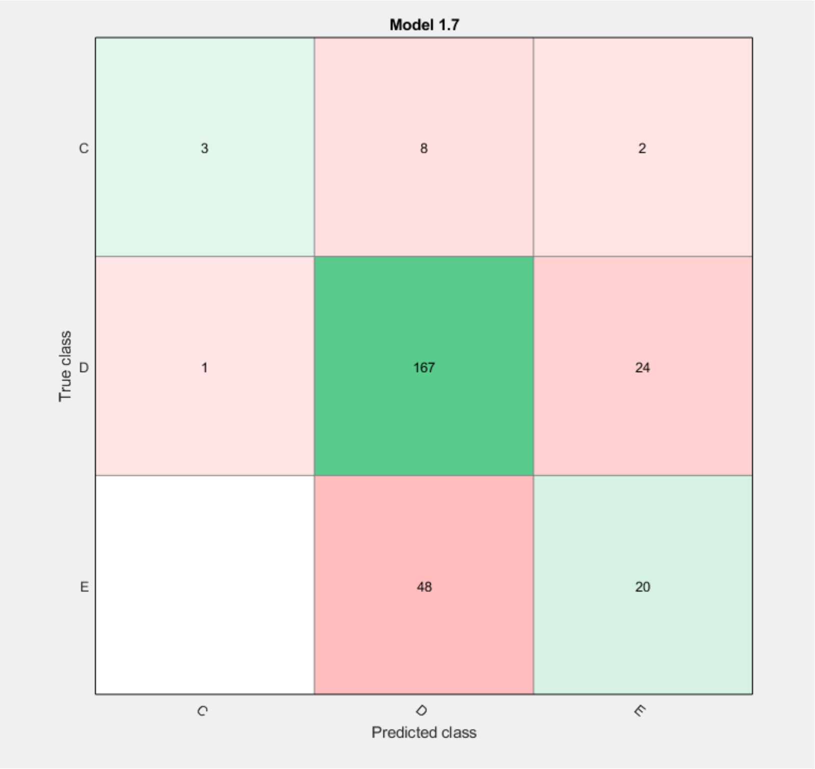
Course Tree



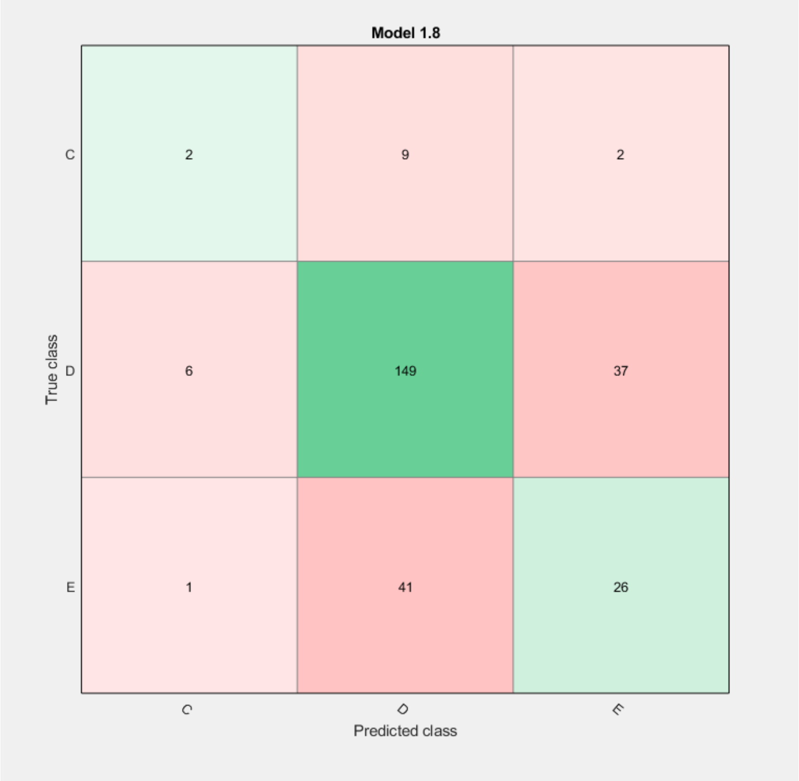
Linear Discriminant



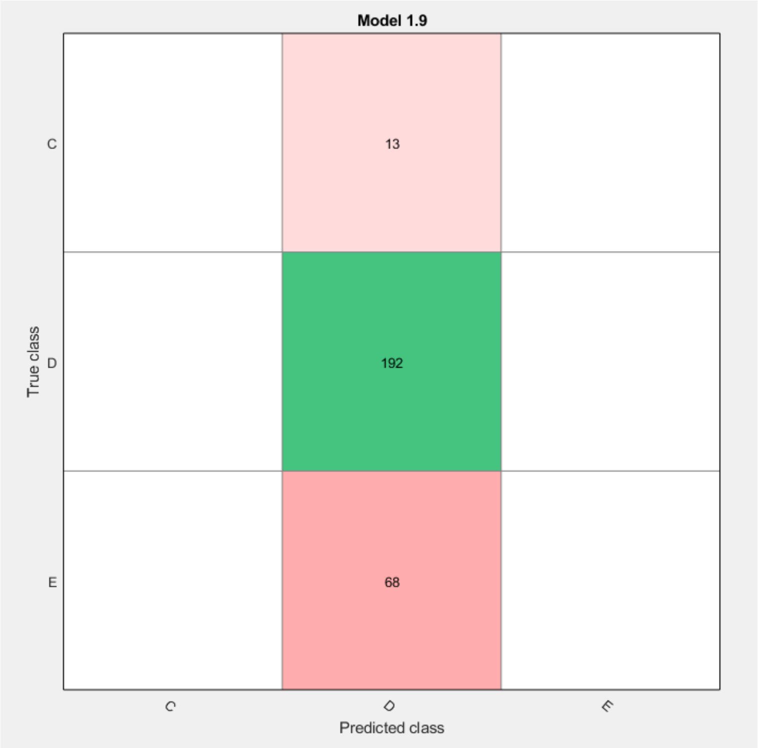
Linear SVM



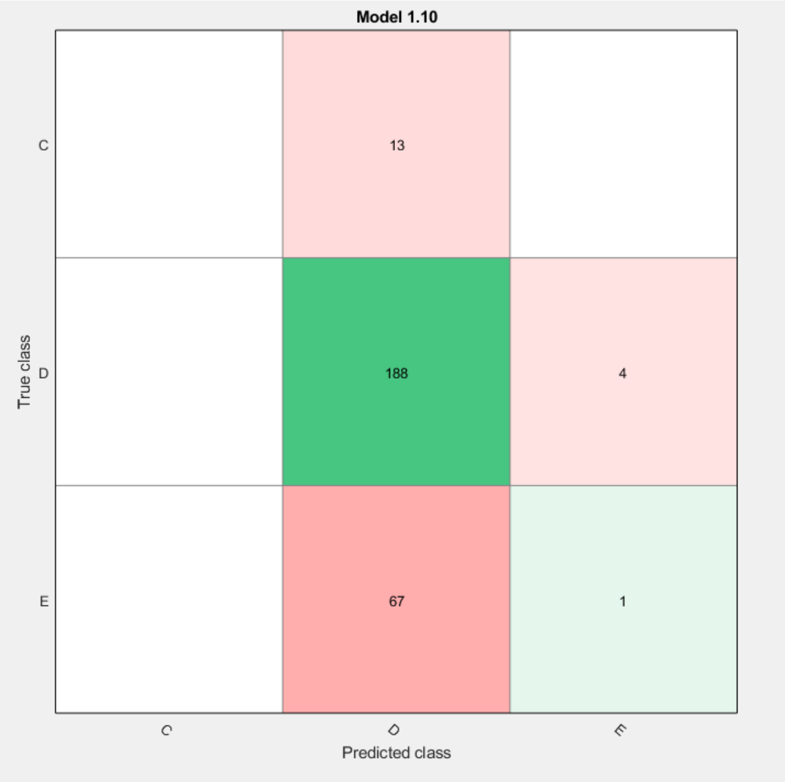
Quadratic SVM



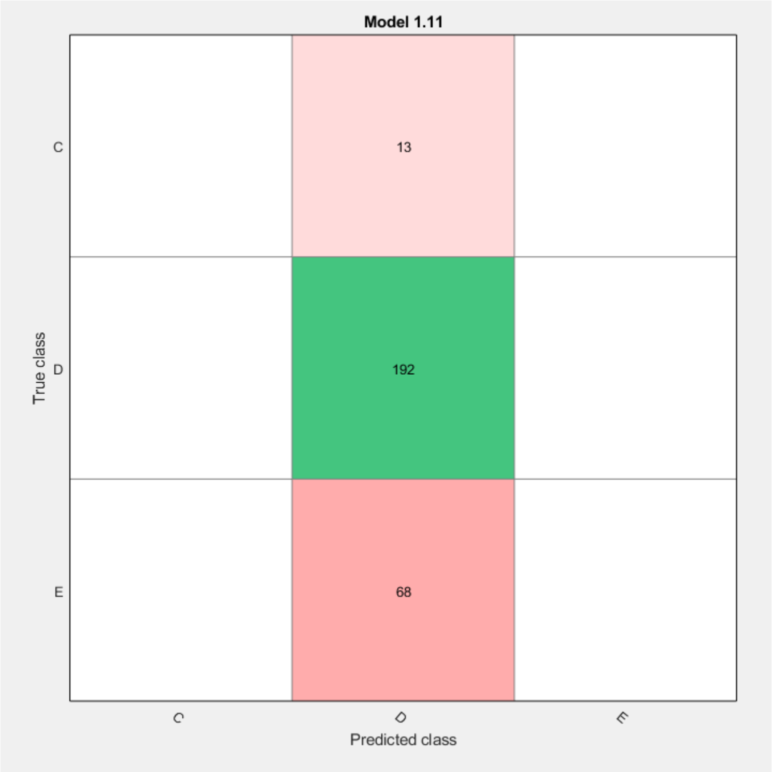
Cubic SVM



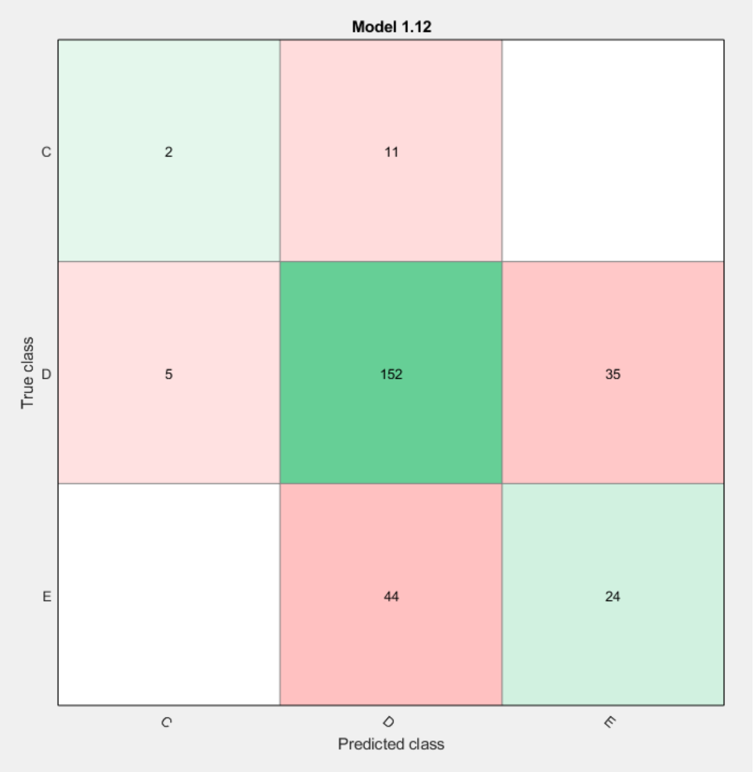
Fine Gaussian SVM



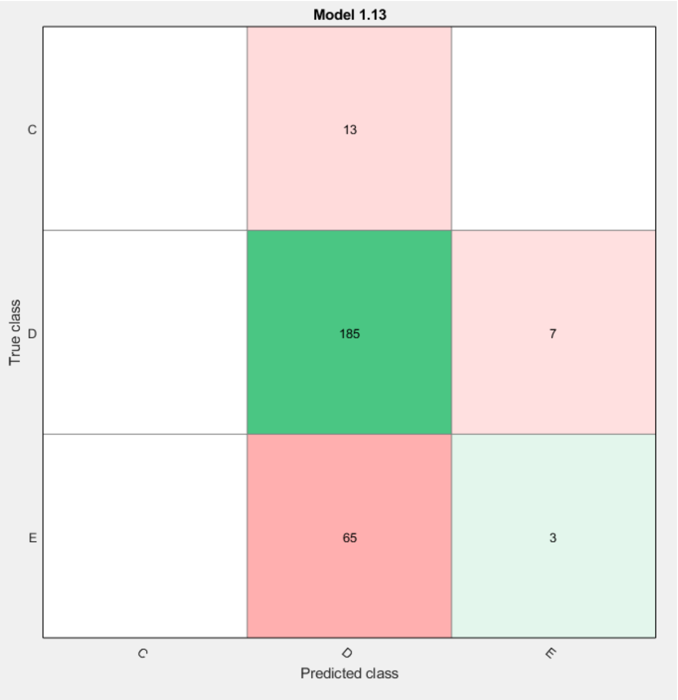
Medium Gaussian SVM



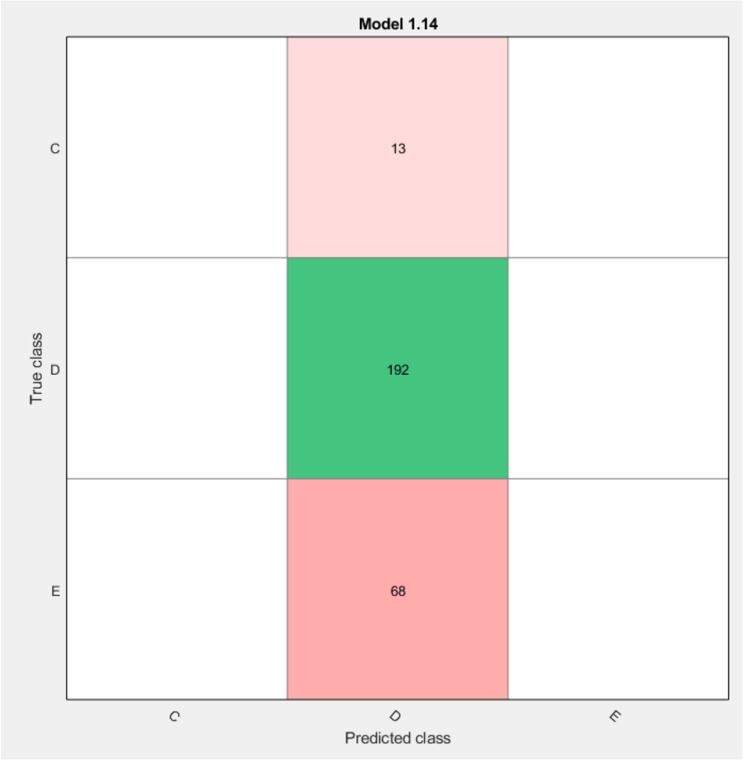
Coarse Gaussian SVM



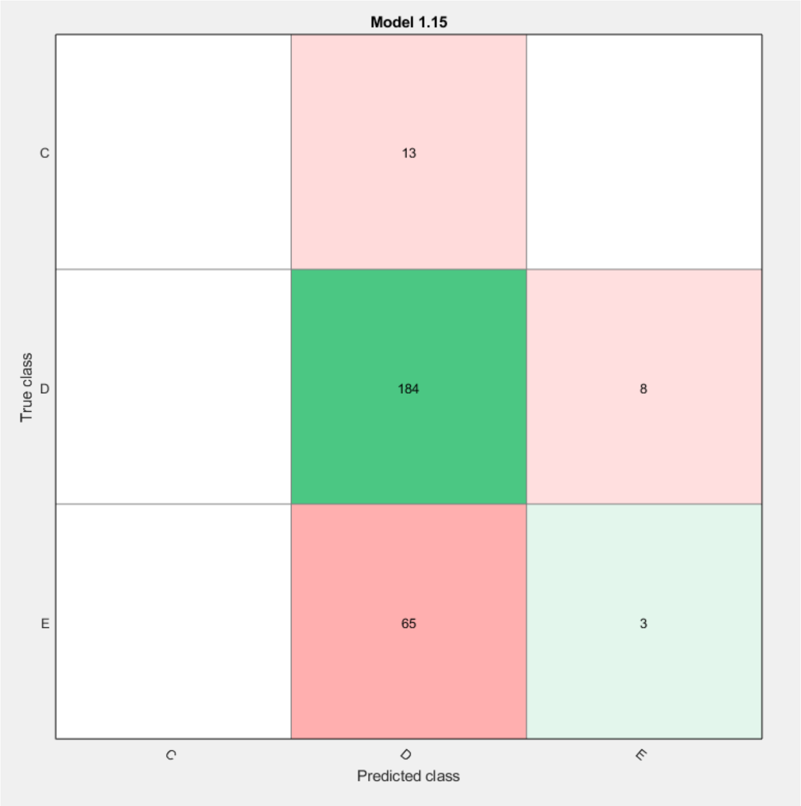
Fine KNN



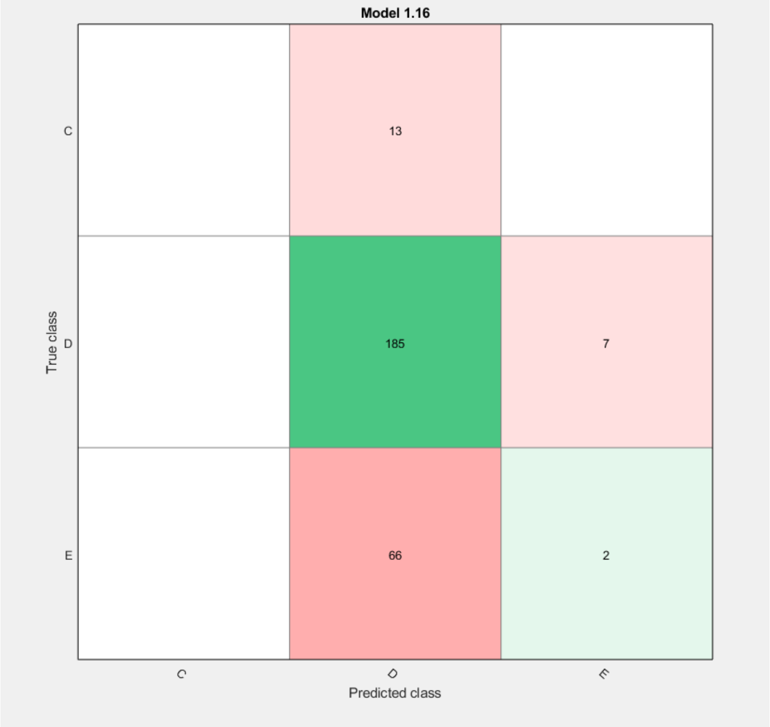
Medium KNN



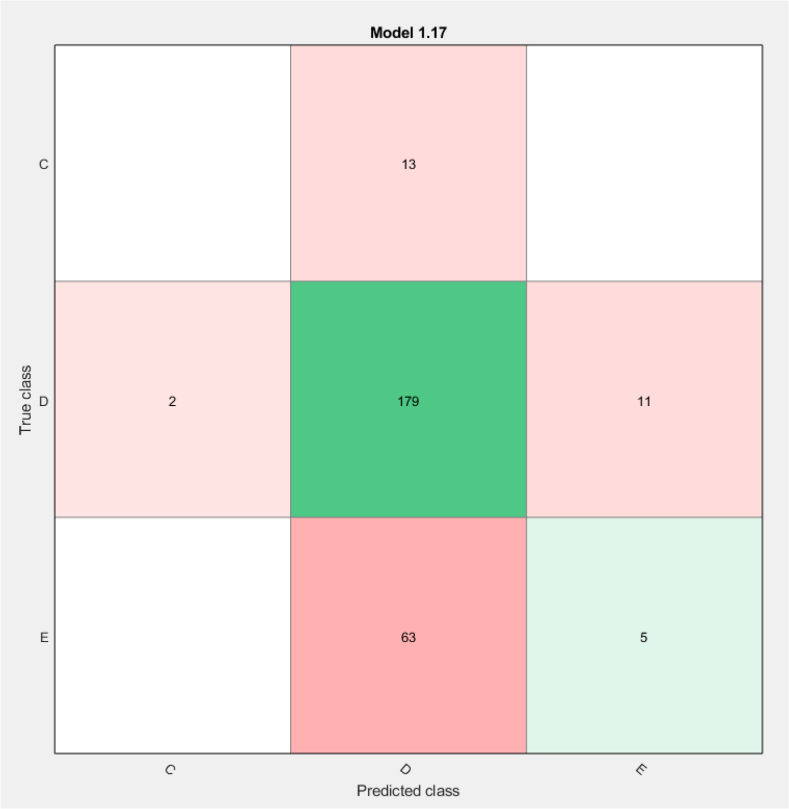
Coarse KNN



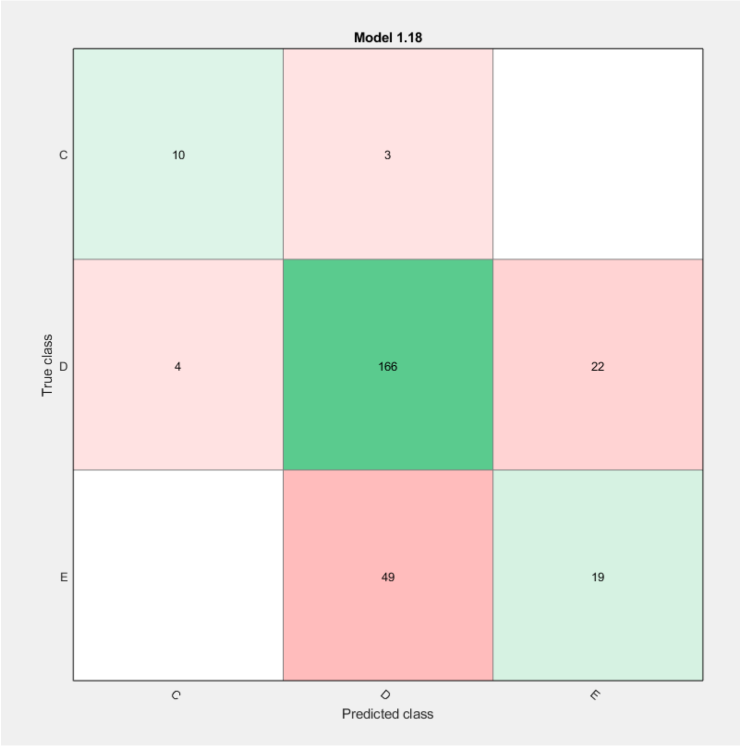
Cosine KNN



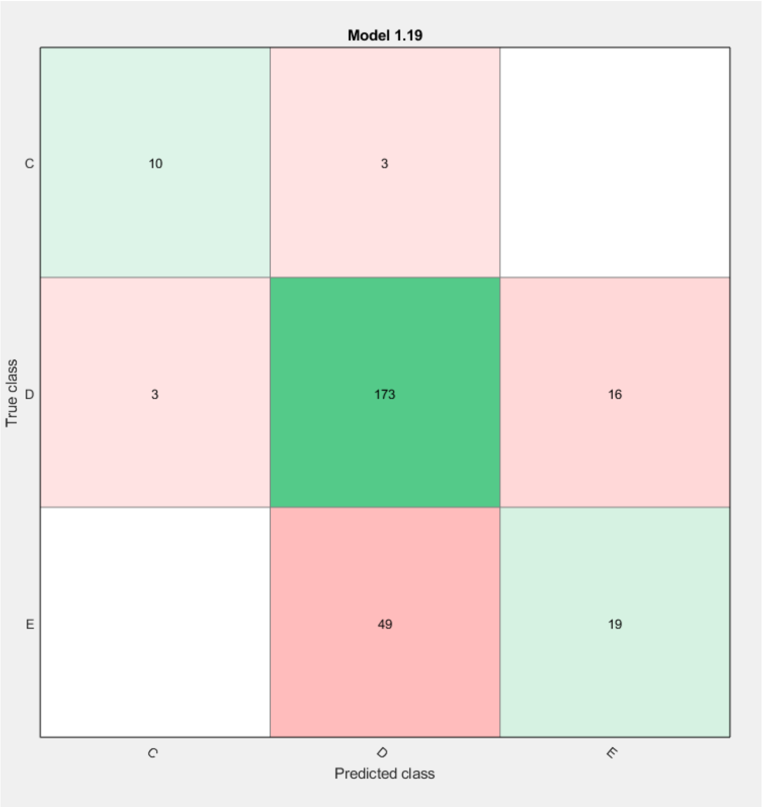
Cubic KNN



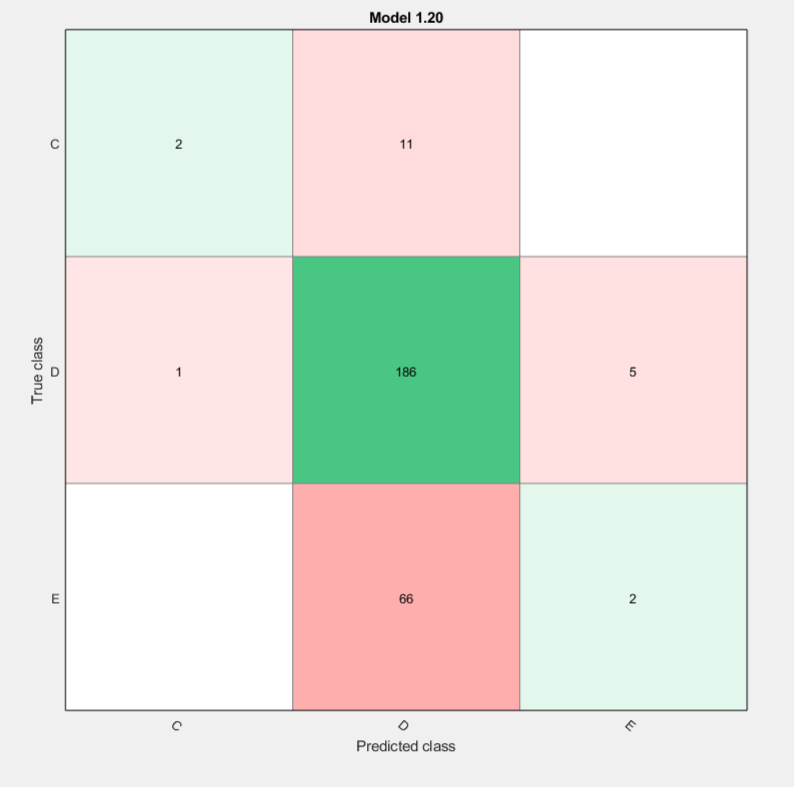
Weighted KNN



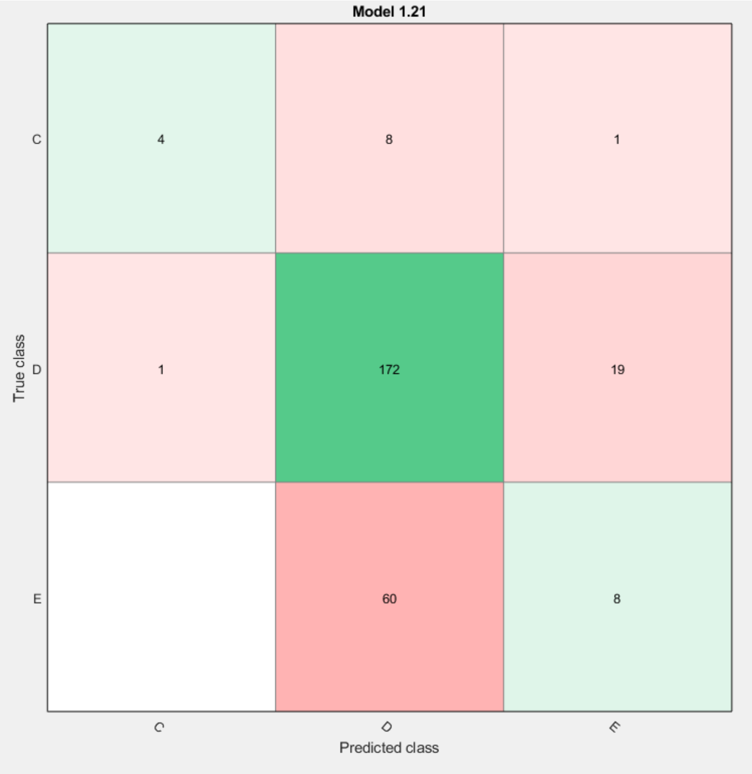
Boosted Trees



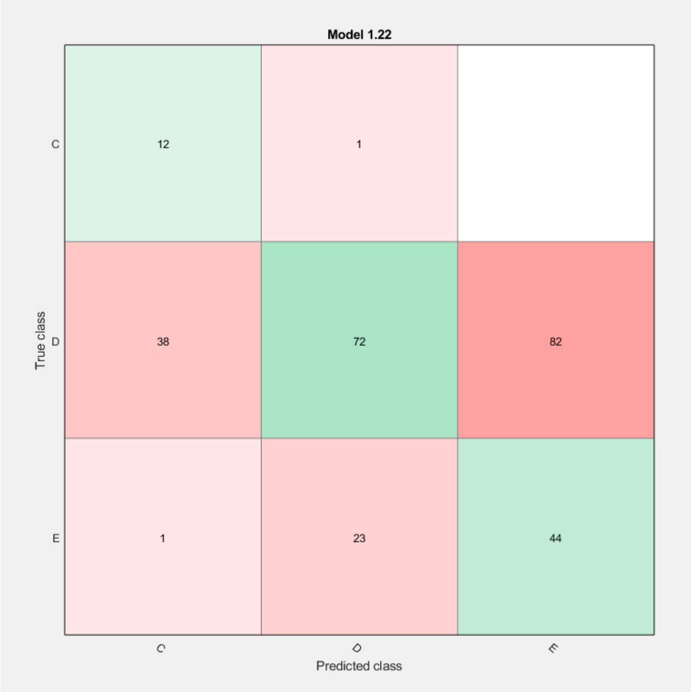
Bagged Trees



Subspace Discriminant

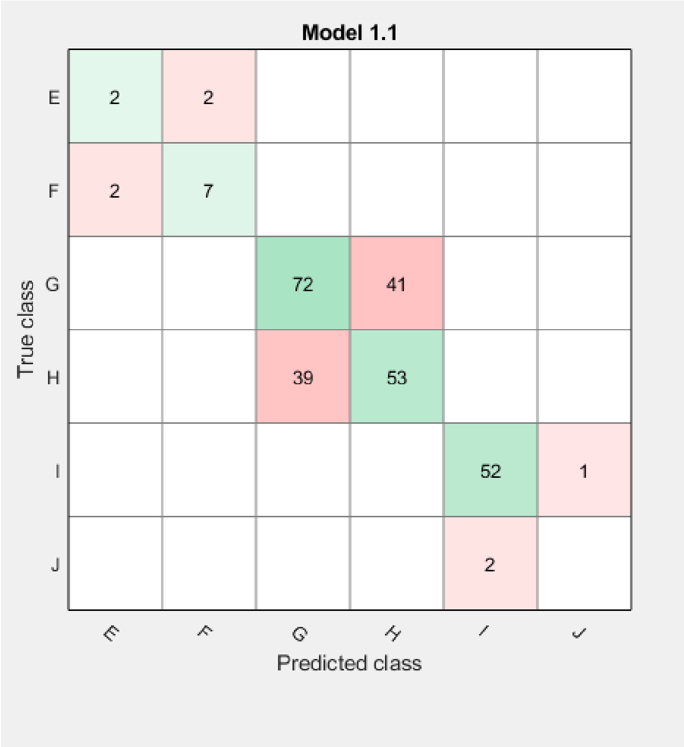


Subspace KNN

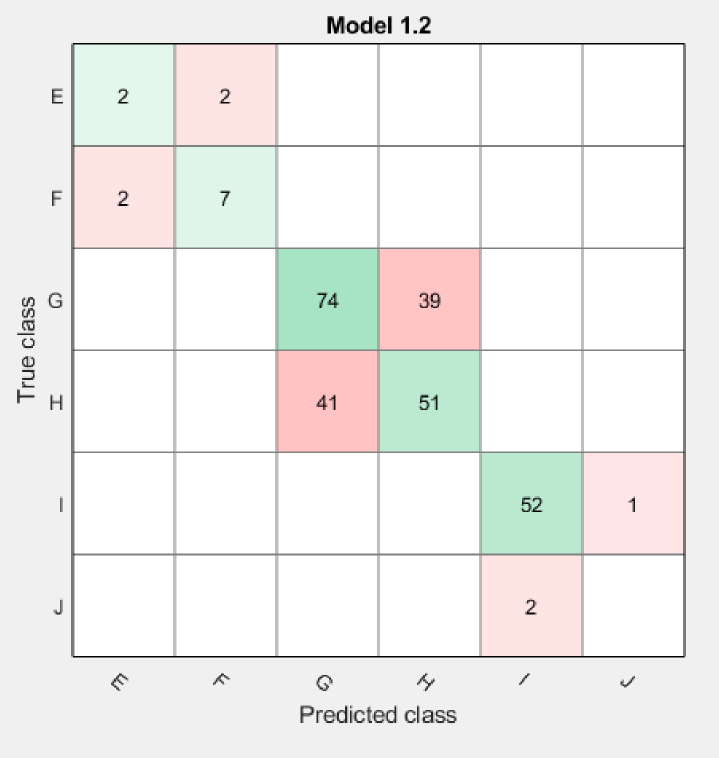


RUSBoosted Trees

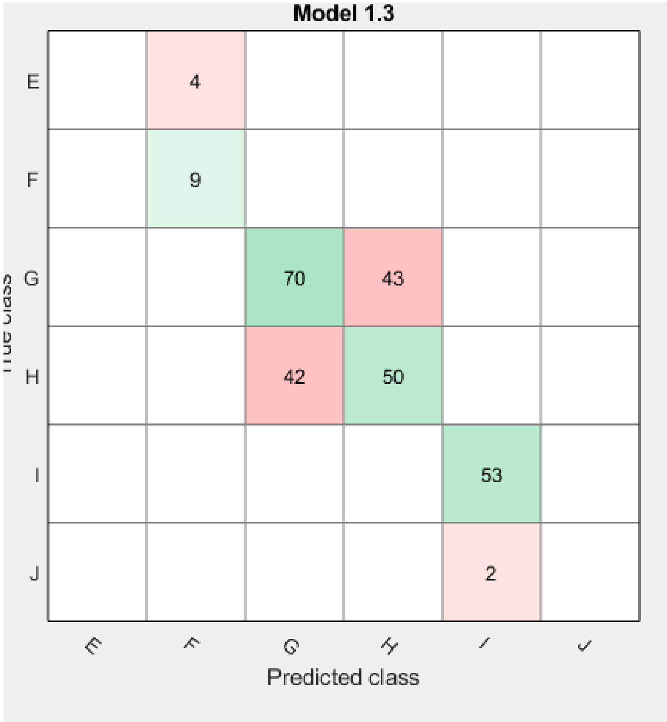
3.2 Bin Size 10



Fine Tree

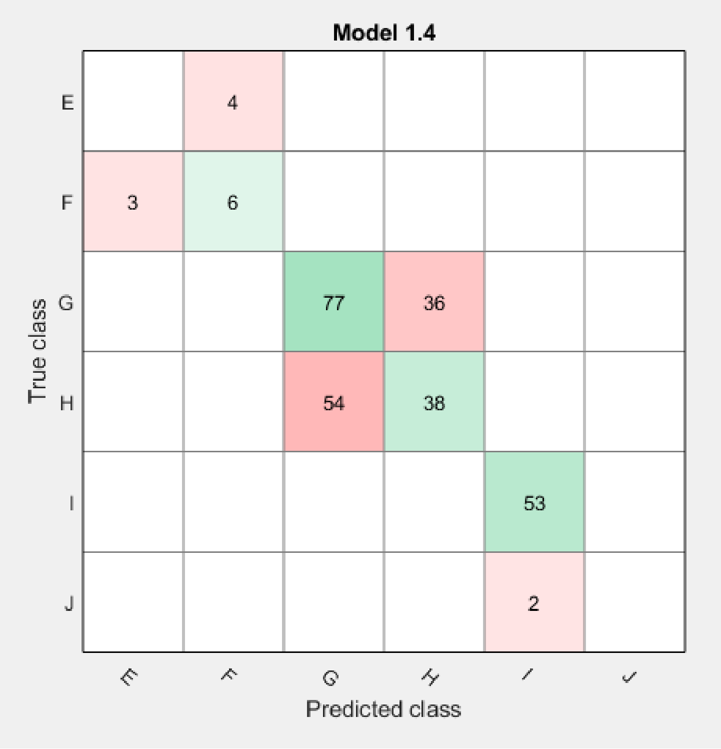


Medium Tree



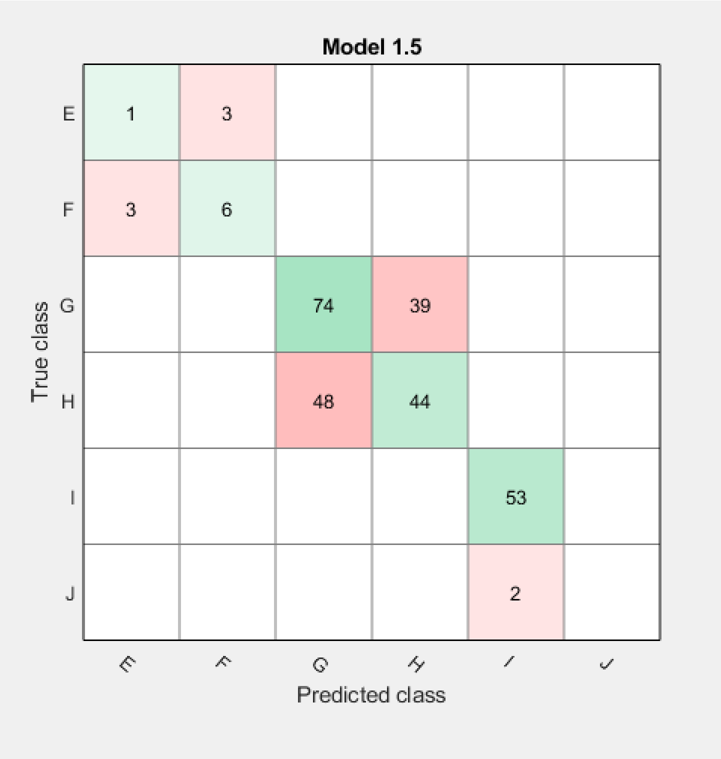
	E	F	G	H	I	J
E		4				
F		9				
G			70	43		
H			42	50		
I					53	
J					2	

Coarse Tree

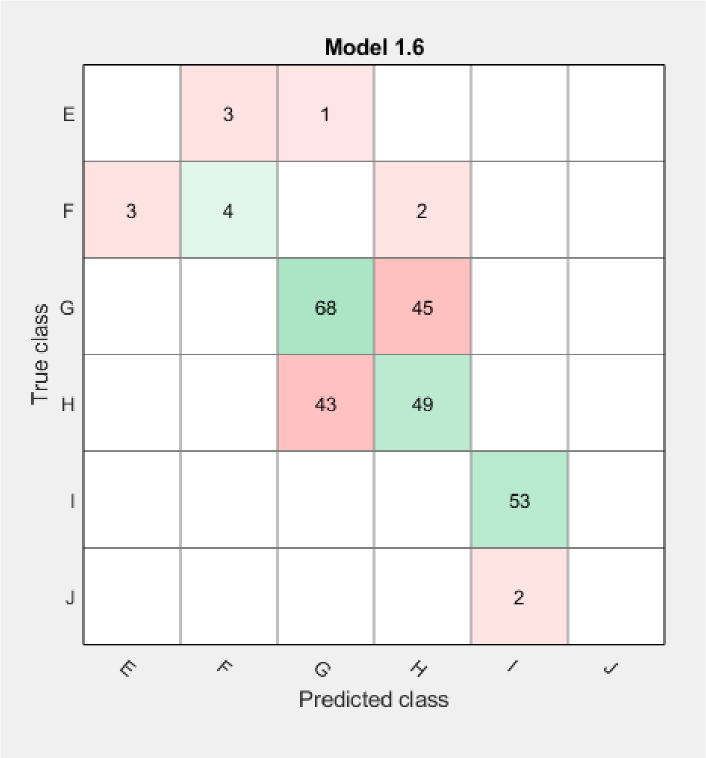


	E	F	G	H	I	J
E		4				
F	3	6				
G			77	36		
H			54	38		
I					53	
J					2	

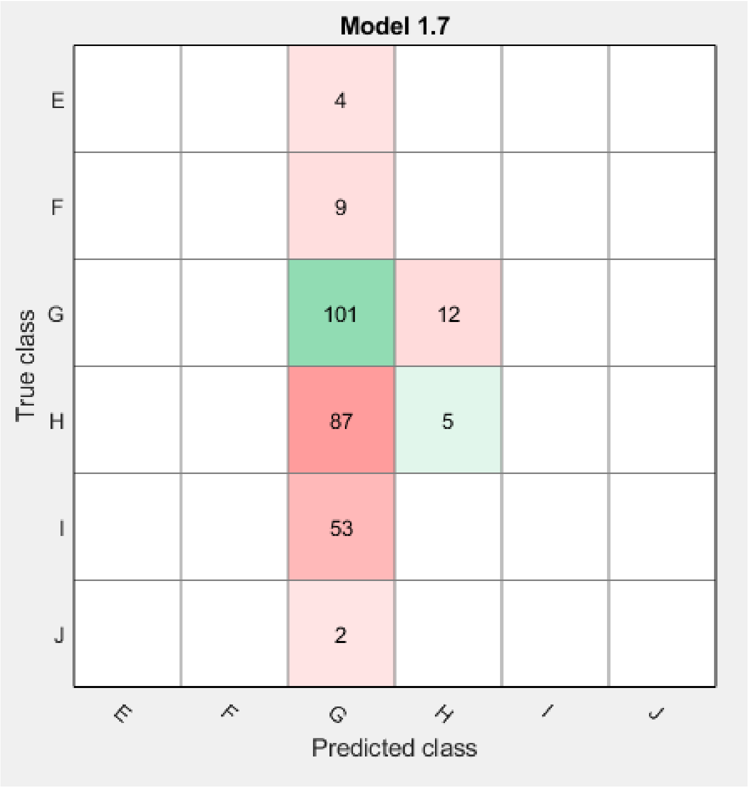
Linear SVM



Quadratic SVM

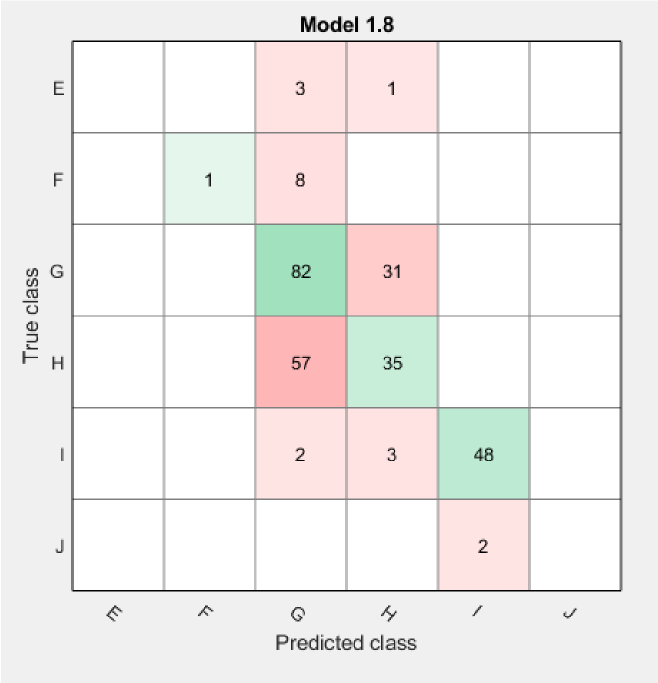


Cubic SVM



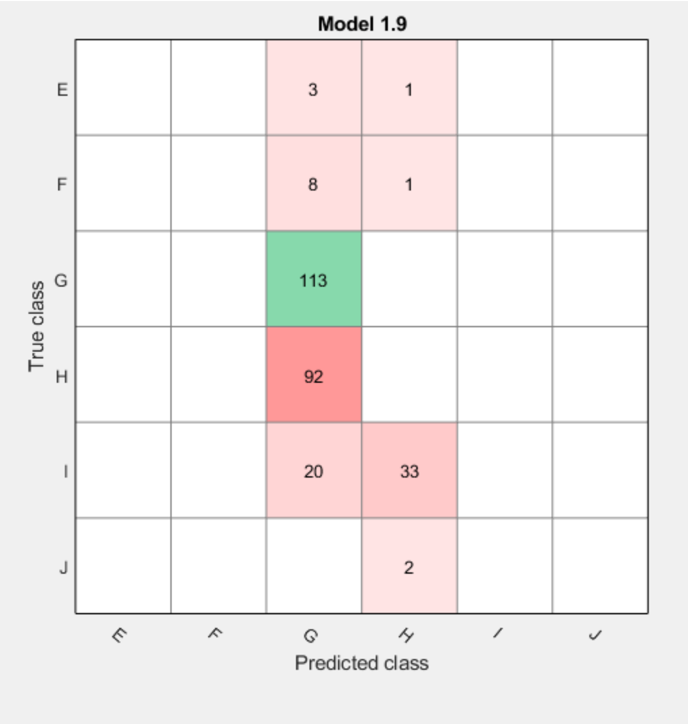
True class \ Predicted class	E	F	G	H	I	J
E			4			
F			9			
G			101	12		
H			87	5		
I			53			
J			2			

Fine Gaussian SVM



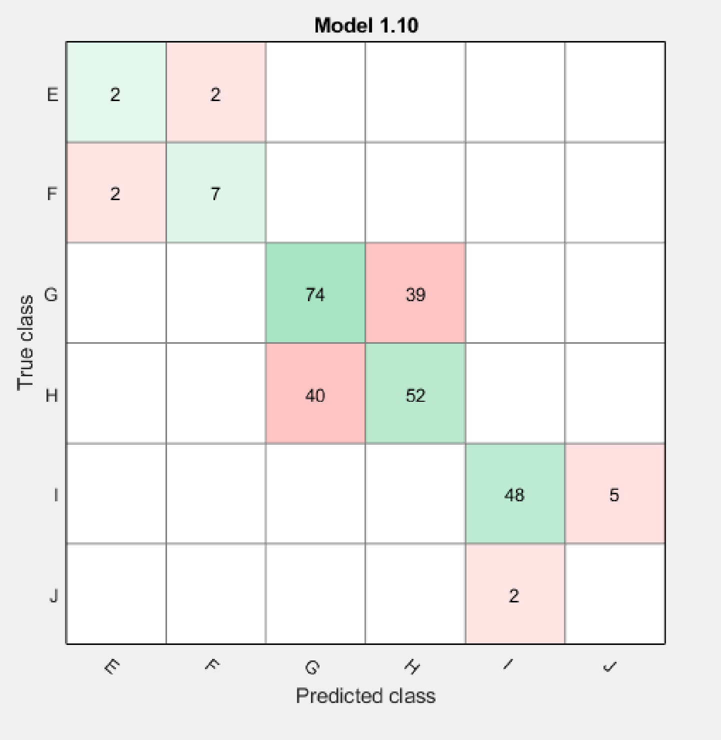
True class \ Predicted class	E	F	G	H	I	J
E			3	1		
F		1	8			
G			82	31		
H			57	35		
I			2	3	48	
J					2	

Medium Gaussian SVM



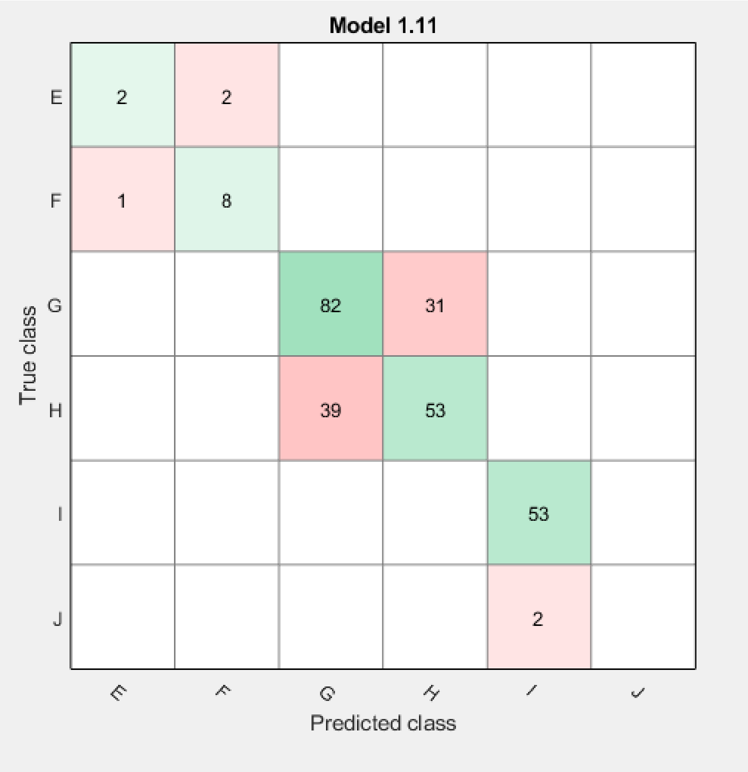
	E	F	G	H	I	J
E			3	1		
F			8	1		
G			113			
H			92			
I			20	33		
J				2		

Coarse Gaussian SVM



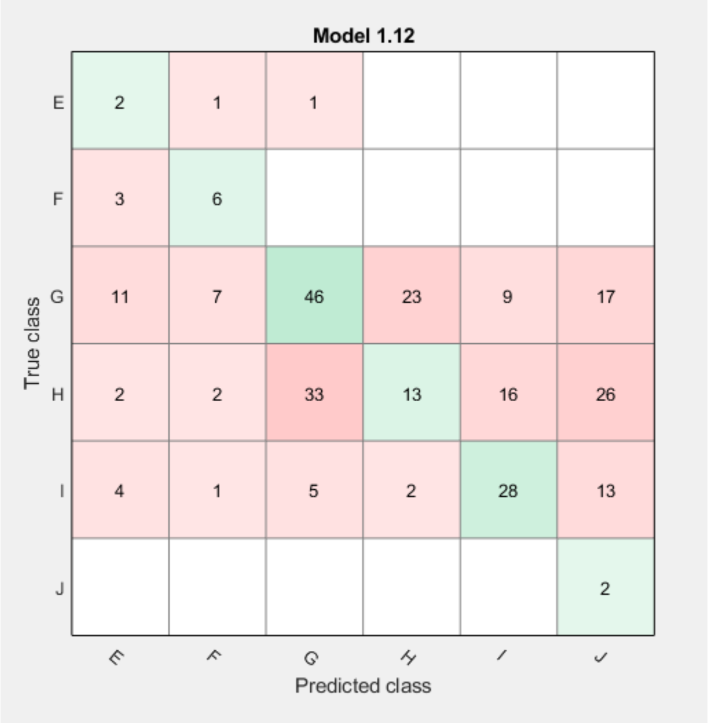
	E	F	G	H	I	J
E	2	2				
F	2	7				
G			74	39		
H			40	52		
I					48	5
J					2	

Boosted Trees



	E	F	G	H	I	J
E	2	2				
F	1	8				
G			82	31		
H			39	53		
I					53	
J					2	

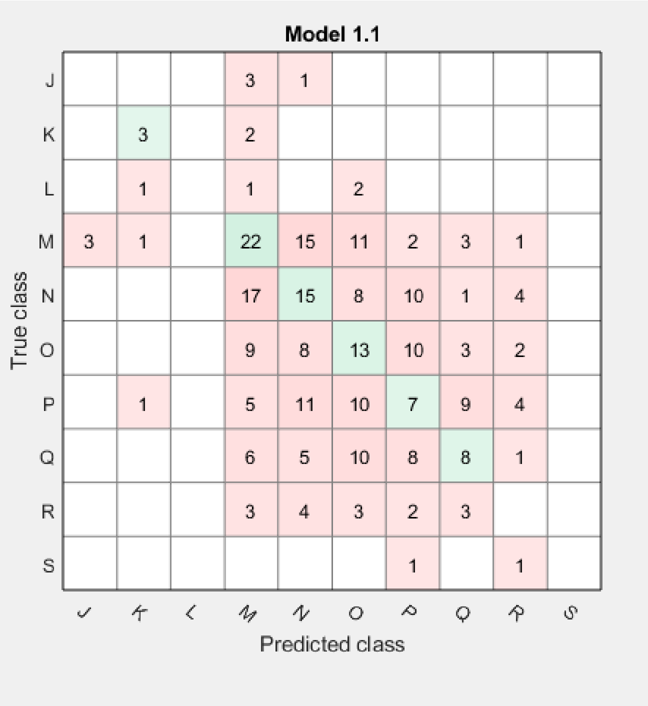
Bagged Trees



	E	F	G	H	I	J
E	2	1	1			
F	3	6				
G	11	7	46	23	9	17
H	2	2	33	13	16	26
I	4	1	5	2	28	13
J						2

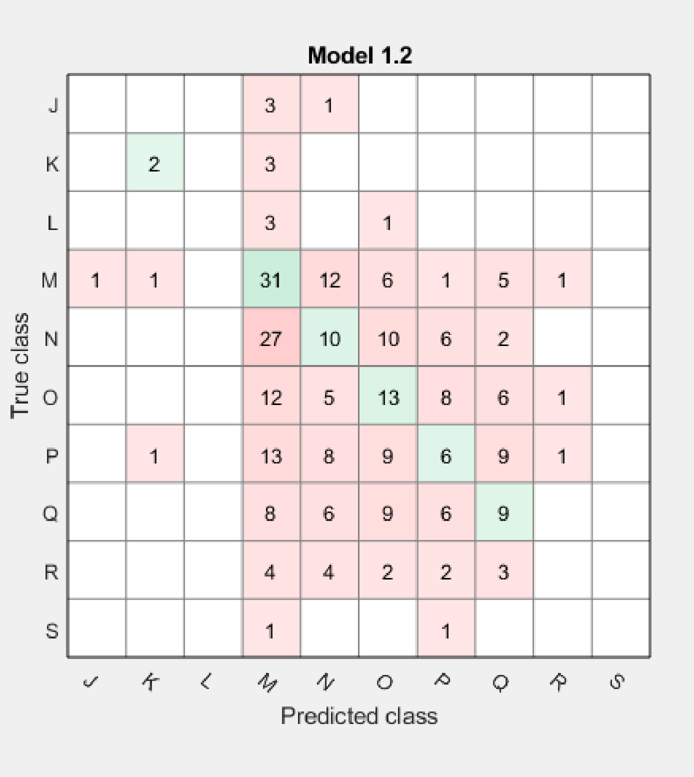
RUSBoosted Trees

3.3 Bin Size 5



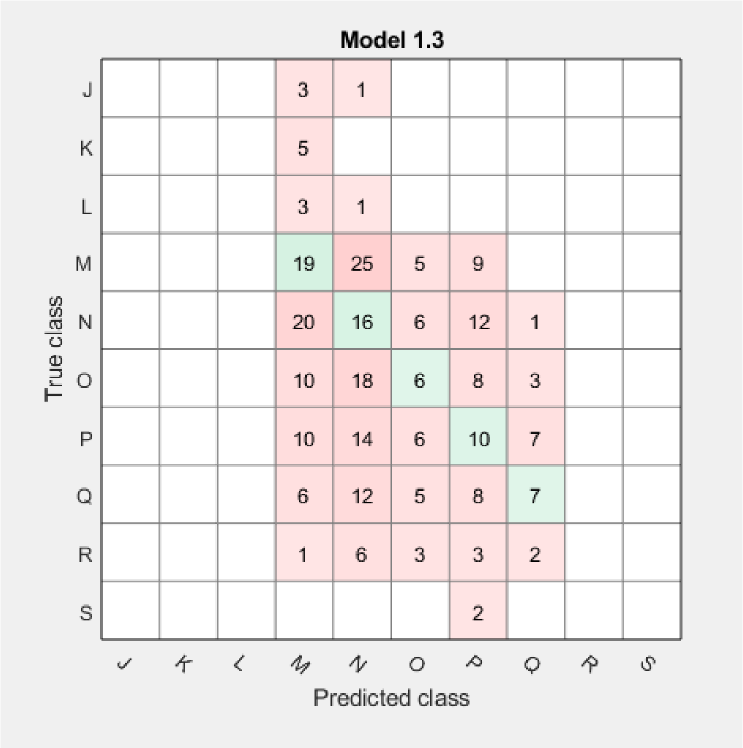
	J	K	L	M	N	O	P	Q	R	S
J				3	1					
K		3		2						
L		1		1		2				
M	3	1		22	15	11	2	3	1	
N				17	15	8	10	1	4	
O				9	8	13	10	3	2	
P		1		5	11	10	7	9	4	
Q				6	5	10	8	8	1	
R				3	4	3	2	3		
S							1		1	

Fine Tree



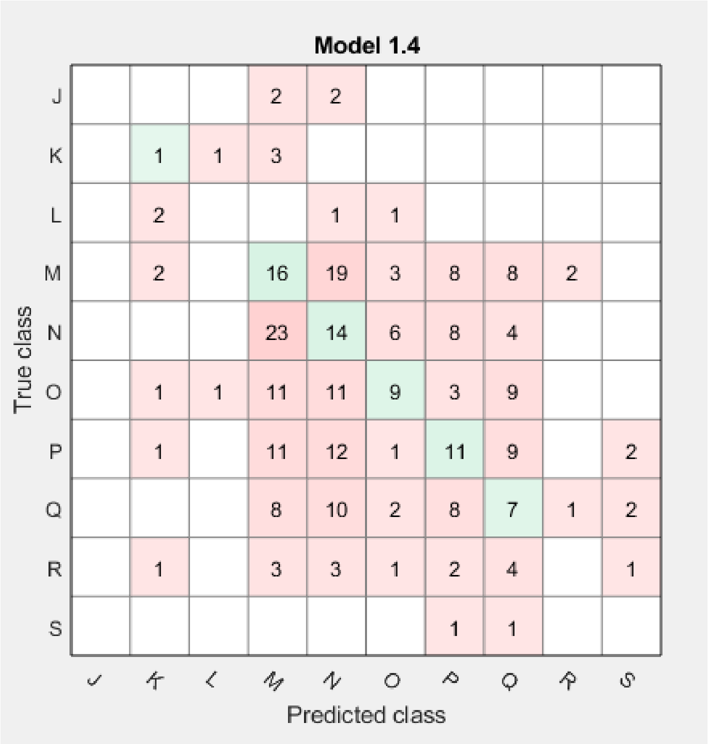
	J	K	L	M	N	O	P	Q	R	S
J				3	1					
K		2		3						
L				3		1				
M	1	1		31	12	6	1	5	1	
N				27	10	10	6	2		
O				12	5	13	8	6	1	
P		1		13	8	9	6	9	1	
Q				8	6	9	6	9		
R				4	4	2	2	3		
S				1			1			

Medium Tree



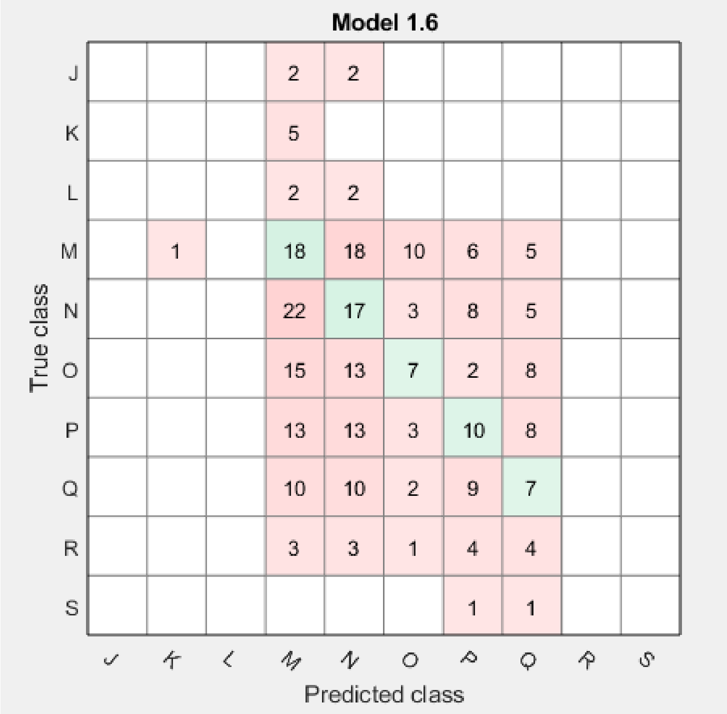
	J	K	L	M	N	O	P	Q	R	S
J				3	1					
K				5						
L				3	1					
M				19	25	5	9			
N				20	16	6	12	1		
O				10	18	6	8	3		
P				10	14	6	10	7		
Q				6	12	5	8	7		
R				1	6	3	3	2		
S							2			

Coarse Tree



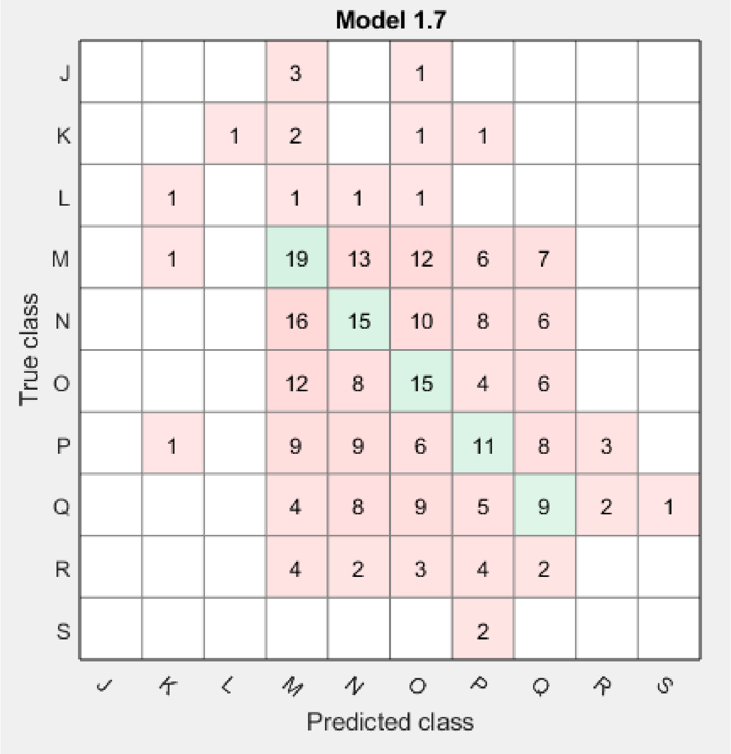
	J	K	L	M	N	O	P	Q	R	S
J				2	2					
K		1	1	3						
L		2			1	1				
M		2		16	19	3	8	8	2	
N				23	14	6	8	4		
O		1	1	11	11	9	3	9		
P		1		11	12	1	11	9		2
Q				8	10	2	8	7	1	2
R		1		3	3	1	2	4		1
S							1	1		

Linear Discriminant



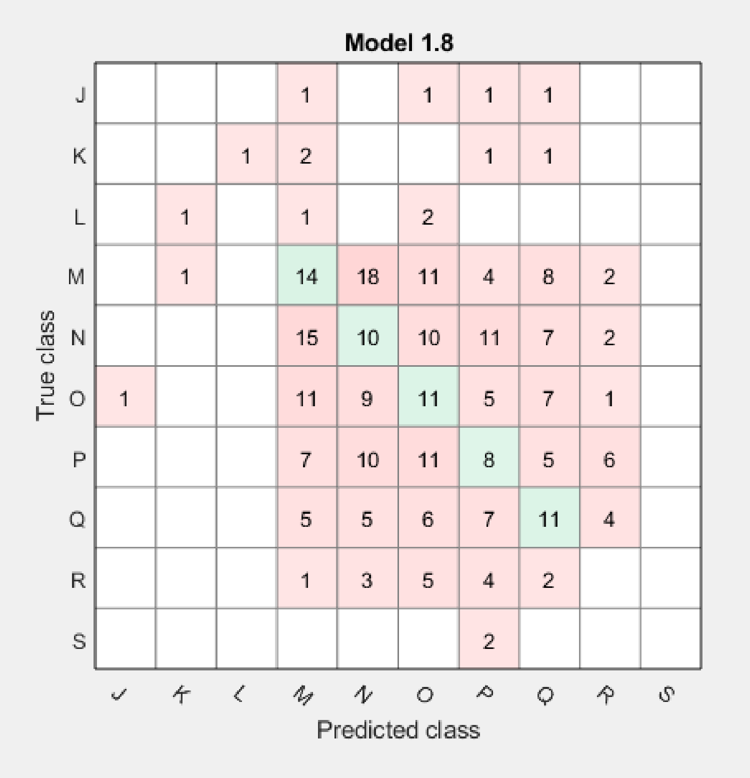
	J	K	L	M	N	O	P	Q	R	S
J				2	2					
K				5						
L				2	2					
M		1		18	18	10	6	5		
N				22	17	3	8	5		
O				15	13	7	2	8		
P				13	13	3	10	8		
Q				10	10	2	9	7		
R				3	3	1	4	4		
S							1	1		

Linear SVM



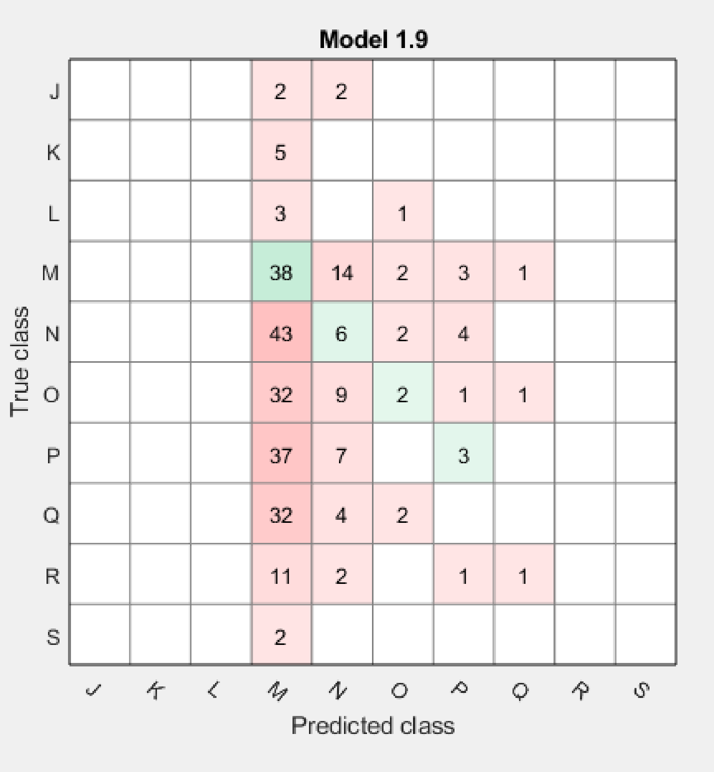
	J	K	L	M	N	O	P	Q	R	S
J				3		1				
K			1	2		1	1			
L		1		1	1	1				
M		1		19	13	12	6	7		
N				16	15	10	8	6		
O				12	8	15	4	6		
P		1		9	9	6	11	8	3	
Q				4	8	9	5	9	2	1
R				4	2	3	4	2		
S							2			

Quadratic SVM



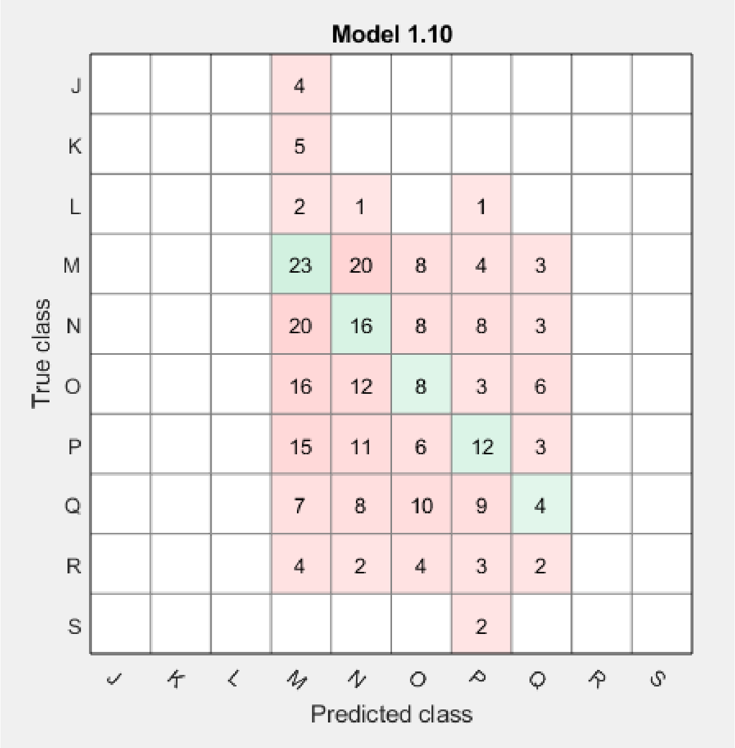
	J	K	L	M	N	O	P	Q	R	S
J				1		1	1	1		
K			1	2			1	1		
L		1		1		2				
M		1		14	18	11	4	8	2	
N				15	10	10	11	7	2	
O	1			11	9	11	5	7	1	
P				7	10	11	8	5	6	
Q				5	5	6	7	11	4	
R				1	3	5	4	2		
S							2			

Cubic SVM



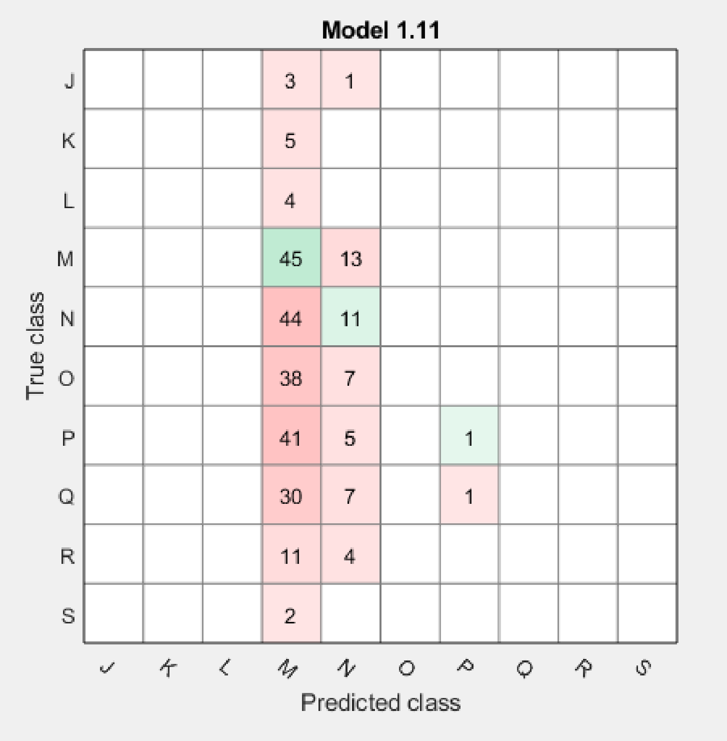
	J	K	L	M	N	O	P	Q	R	S
J				2	2					
K				5						
L				3		1				
M				38	14	2	3	1		
N				43	6	2	4			
O				32	9	2	1	1		
P				37	7		3			
Q				32	4	2				
R				11	2		1	1		
S				2						

Fine Gaussian SVM


Model 1.10
True class
Predicted class

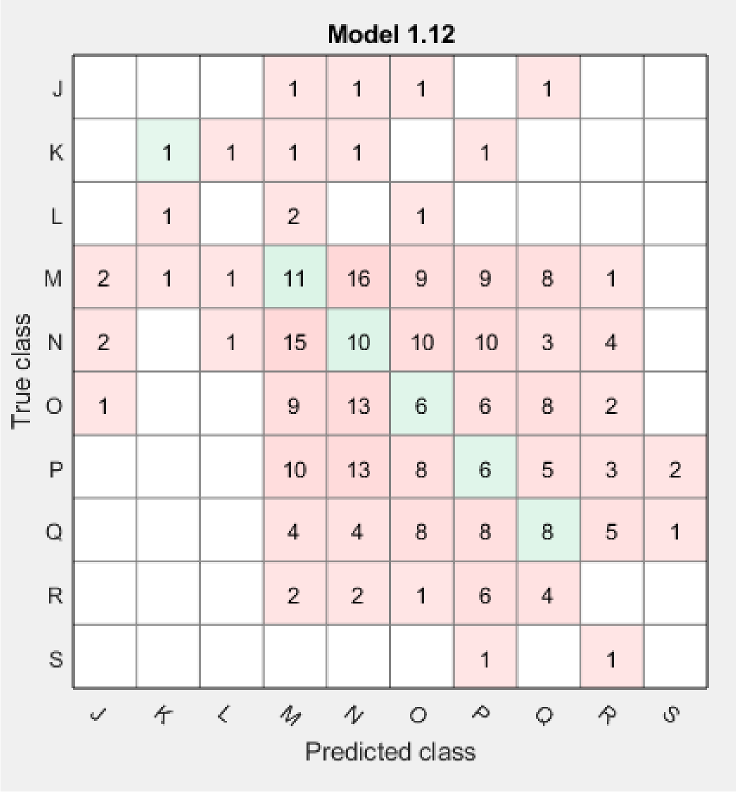
	J	K	L	M	N	O	P	Q	R	S
J				4						
K				5						
L				2	1		1			
M				23	20	8	4	3		
N				20	16	8	8	3		
O				16	12	8	3	6		
P				15	11	6	12	3		
Q				7	8	10	9	4		
R				4	2	4	3	2		
S							2			

Medium Gaussian SVM


Model 1.11
True class
Predicted class

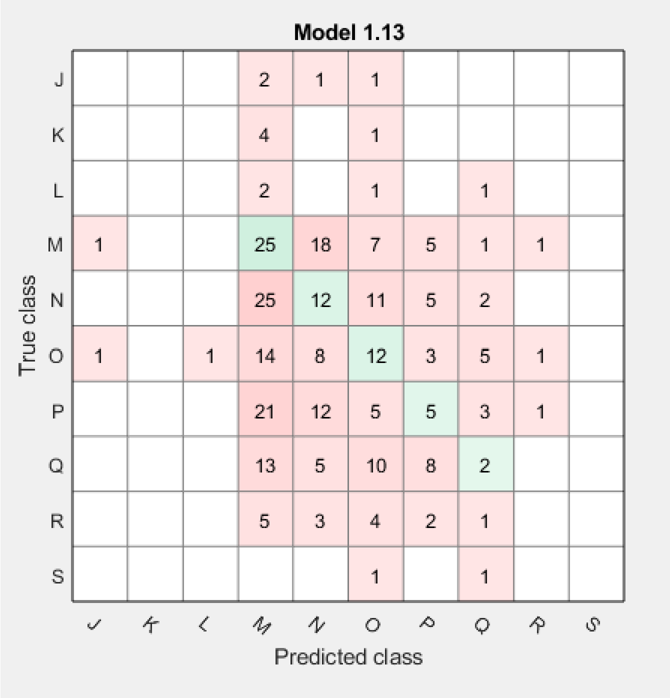
	J	K	L	M	N	O	P	Q	R	S
J				3	1					
K				5						
L				4						
M				45	13					
N				44	11					
O				38	7					
P				41	5		1			
Q				30	7		1			
R				11	4					
S				2						

Coarse Gaussian SVM



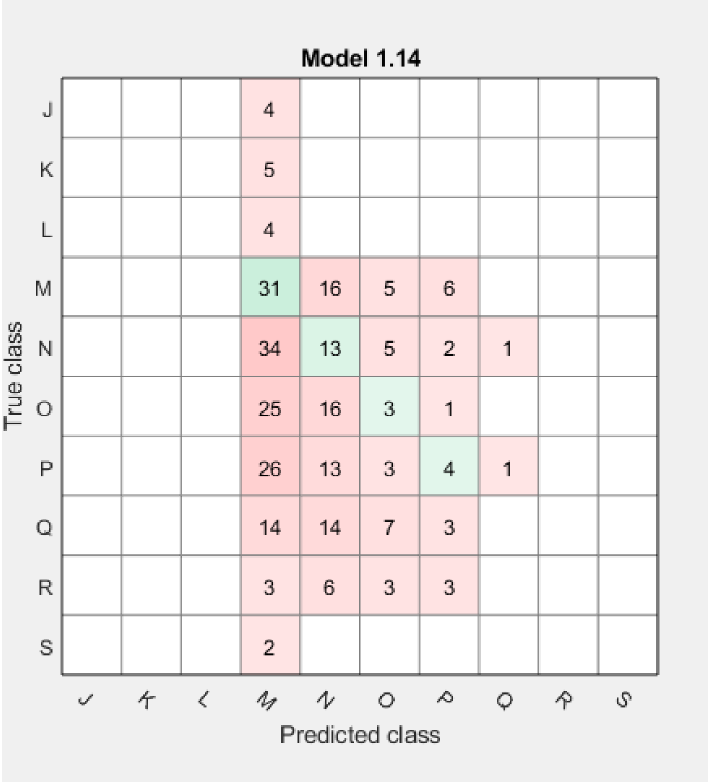
	J	K	L	M	N	O	P	Q	R	S
J				1	1	1		1		
K		1	1	1	1		1			
L		1		2		1				
M	2	1	1	11	16	9	9	8	1	
N	2		1	15	10	10	10	3	4	
O	1			9	13	6	6	8	2	
P				10	13	8	6	5	3	2
Q				4	4	8	8	8	5	1
R				2	2	1	6	4		
S							1		1	

Fine KNN



	J	K	L	M	N	O	P	Q	R	S
J				2	1	1				
K				4		1				
L				2		1		1		
M	1			25	18	7	5	1	1	
N				25	12	11	5	2		
O	1		1	14	8	12	3	5	1	
P				21	12	5	5	3	1	
Q				13	5	10	8	2		
R				5	3	4	2	1		
S						1		1		

Medium KNN

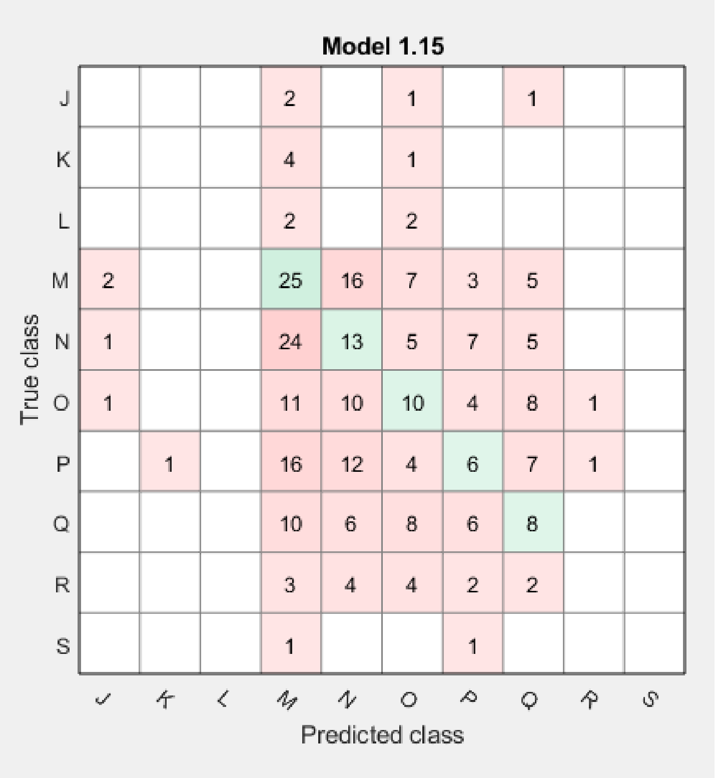


Model 1.14

True class
Predicted class

	J	K	L	M	N	O	P	Q	R	S
J				4						
K				5						
L				4						
M				31	16	5	6			
N				34	13	5	2	1		
O				25	16	3	1			
P				26	13	3	4	1		
Q				14	14	7	3			
R				3	6	3	3			
S				2						

Coarse KNN

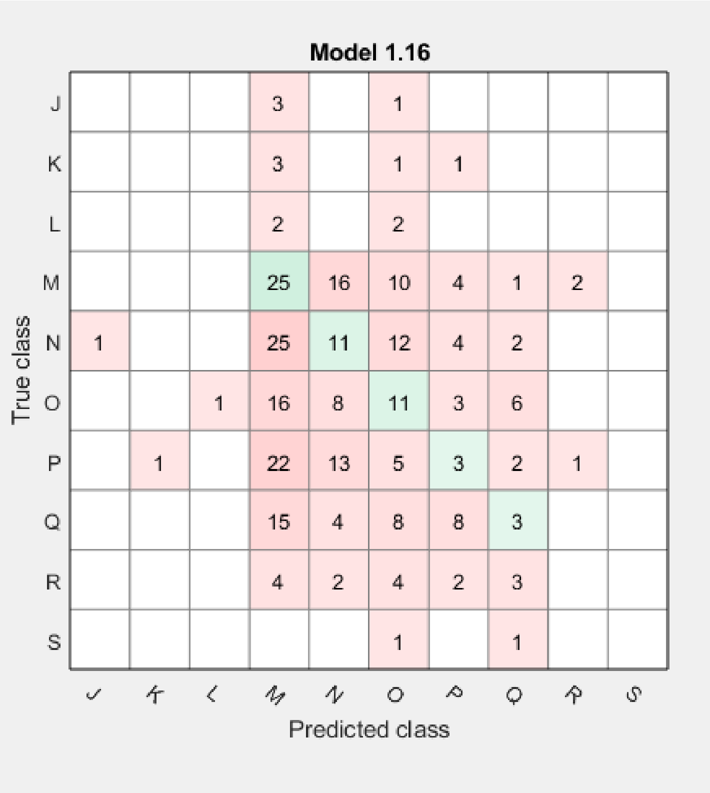


Model 1.15

True class
Predicted class

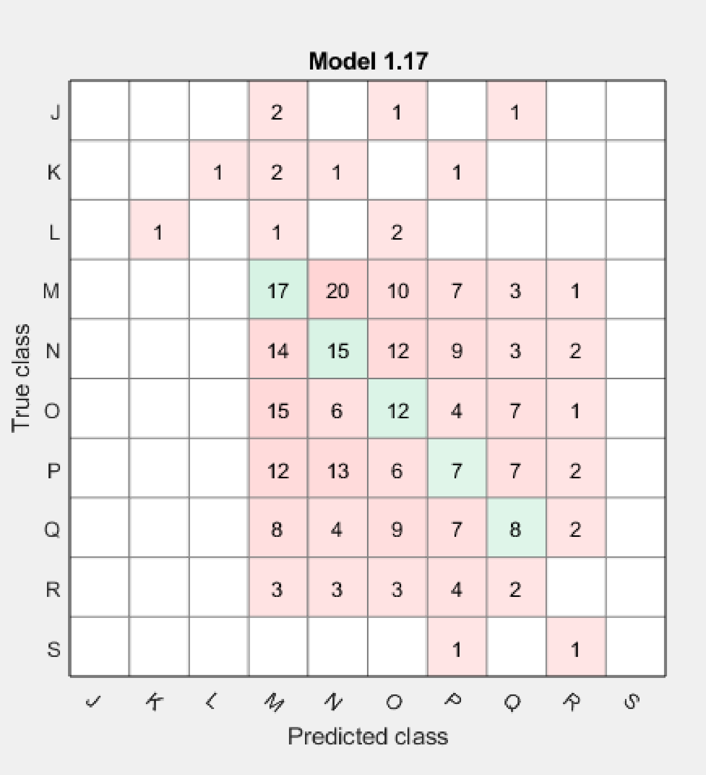
	J	K	L	M	N	O	P	Q	R	S
J				2		1		1		
K				4		1				
L				2		2				
M	2			25	16	7	3	5		
N	1			24	13	5	7	5		
O	1			11	10	10	4	8	1	
P		1		16	12	4	6	7	1	
Q				10	6	8	6	8		
R				3	4	4	2	2		
S				1			1			

Cosine KNN



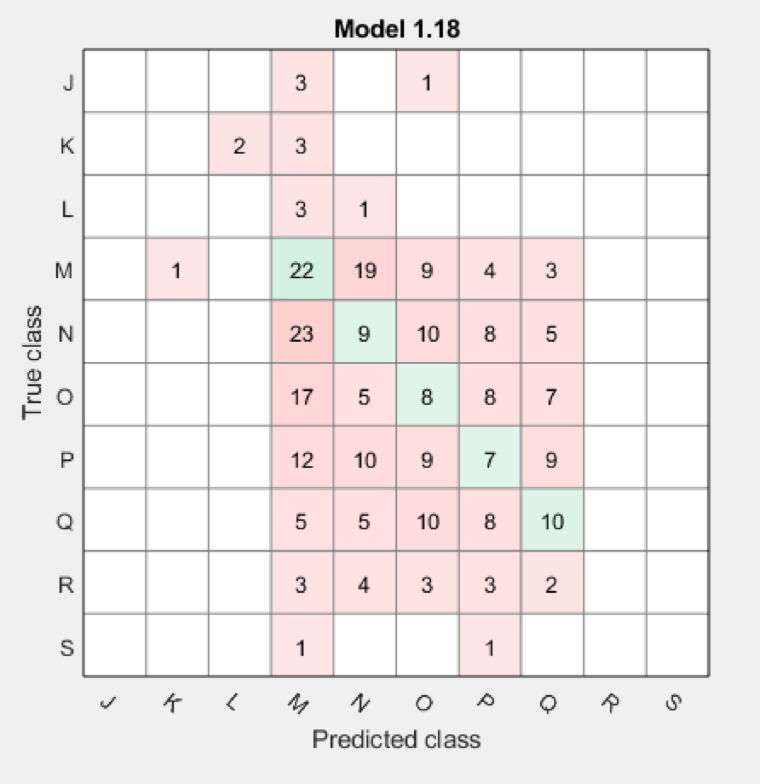
	J	K	L	M	N	O	P	Q	R	S
J				3		1				
K				3		1	1			
L				2		2				
M				25	16	10	4	1	2	
N	1			25	11	12	4	2		
O			1	16	8	11	3	6		
P		1		22	13	5	3	2	1	
Q				15	4	8	8	3		
R				4	2	4	2	3		
S						1		1		

Cubic KNN



	J	K	L	M	N	O	P	Q	R	S
J				2		1		1		
K			1	2	1		1			
L		1		1		2				
M				17	20	10	7	3	1	
N				14	15	12	9	3	2	
O				15	6	12	4	7	1	
P				12	13	6	7	7	2	
Q				8	4	9	7	8	2	
R				3	3	3	4	2		
S							1		1	

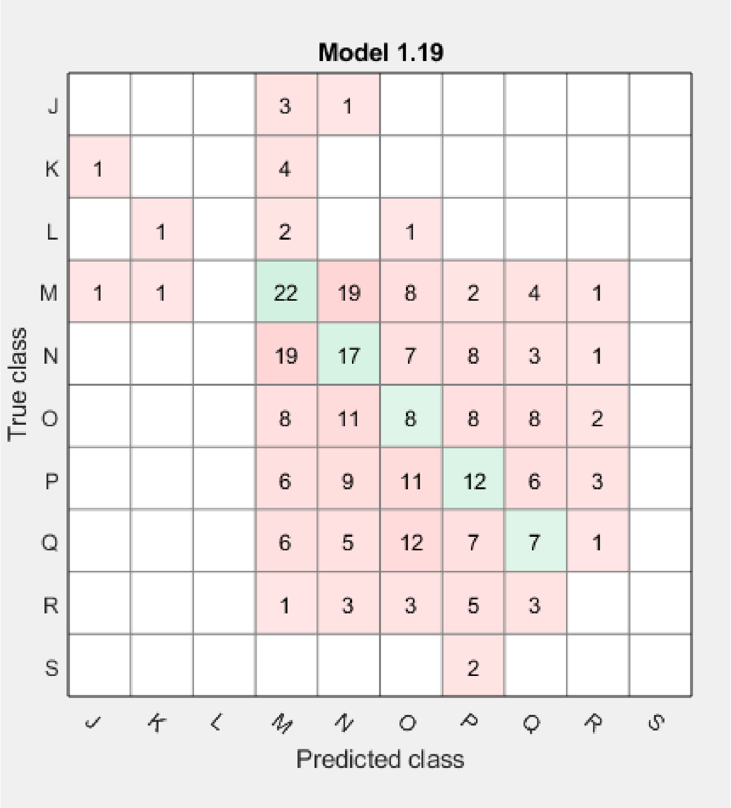
Weighted KNN



Model 1.18

	J	K	L	M	N	O	P	Q	R	S
J				3		1				
K			2	3						
L				3	1					
M		1		22	19	9	4	3		
N				23	9	10	8	5		
O				17	5	8	8	7		
P				12	10	9	7	9		
Q				5	5	10	8	10		
R				3	4	3	3	2		
S				1			1			

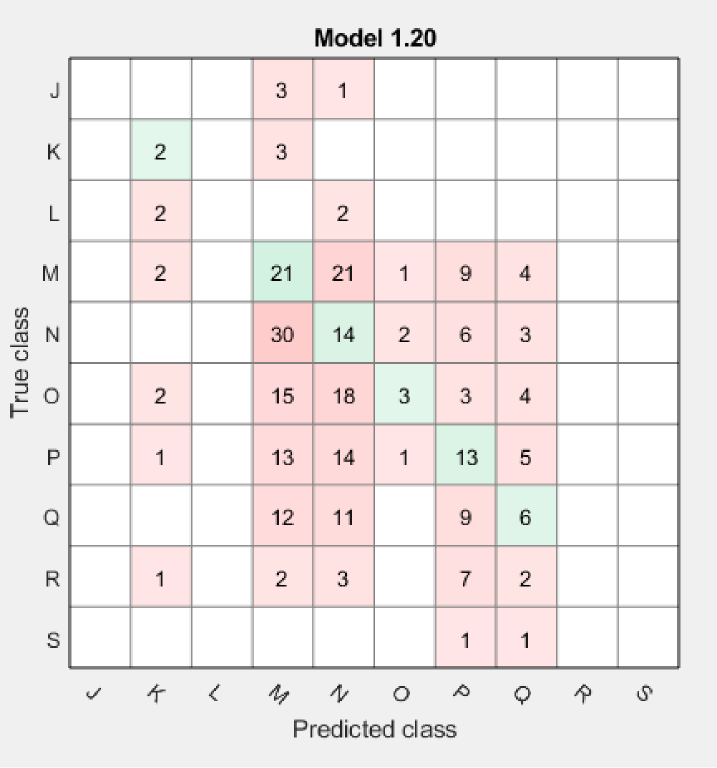
Boosted Trees



Model 1.19

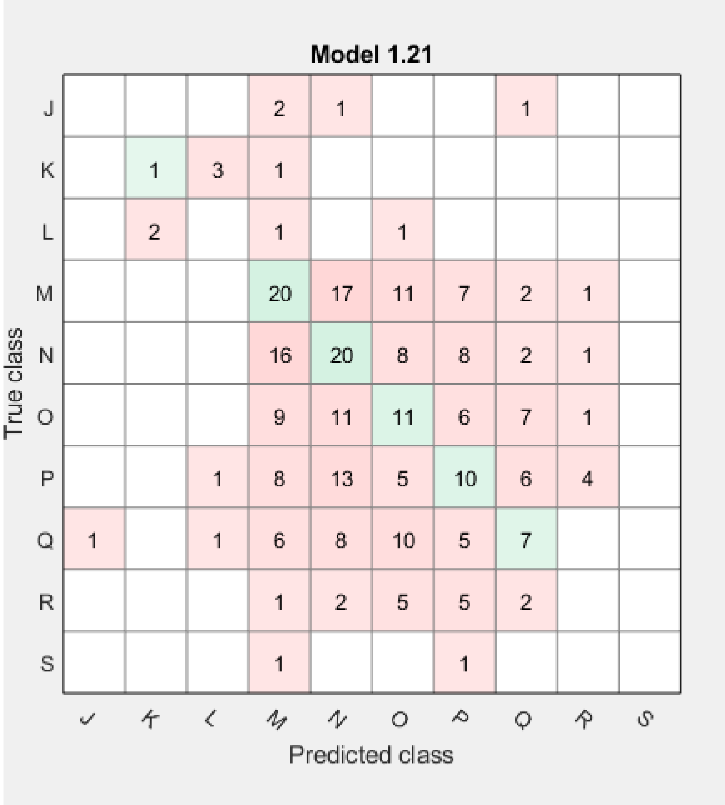
	J	K	L	M	N	O	P	Q	R	S
J				3	1					
K	1			4						
L		1		2		1				
M	1	1		22	19	8	2	4	1	
N				19	17	7	8	3	1	
O				8	11	8	8	8	2	
P				6	9	11	12	6	3	
Q				6	5	12	7	7	1	
R				1	3	3	5	3		
S							2			

Bagged Trees



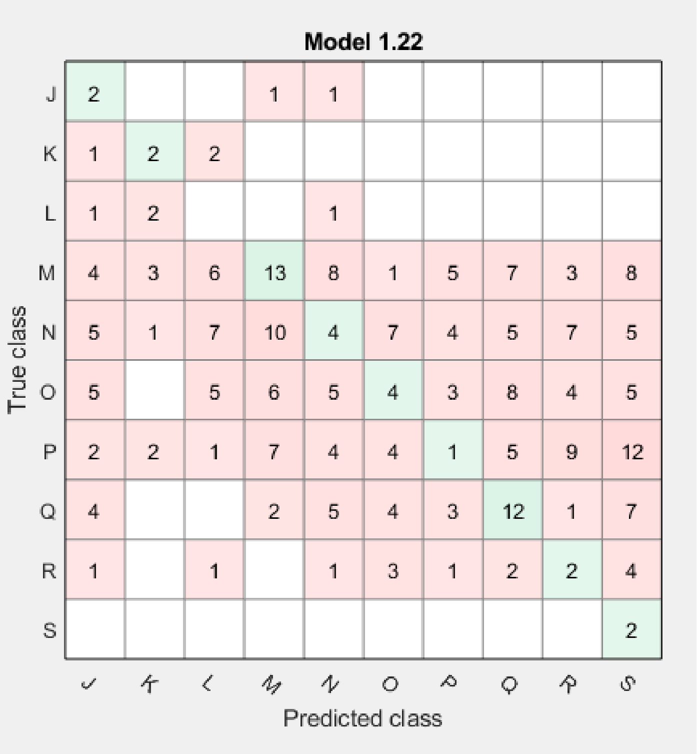
	J	K	L	M	N	O	P	Q	R	S
J				3	1					
K		2		3						
L		2			2					
M		2		21	21	1	9	4		
N				30	14	2	6	3		
O		2		15	18	3	3	4		
P		1		13	14	1	13	5		
Q				12	11		9	6		
R		1		2	3		7	2		
S							1	1		

Subspace Discriminant



	J	K	L	M	N	O	P	Q	R	S
J				2	1			1		
K		1	3	1						
L		2		1		1				
M				20	17	11	7	2	1	
N				16	20	8	8	2	1	
O				9	11	11	6	7	1	
P			1	8	13	5	10	6	4	
Q	1		1	6	8	10	5	7		
R				1	2	5	5	2		
S				1			1			

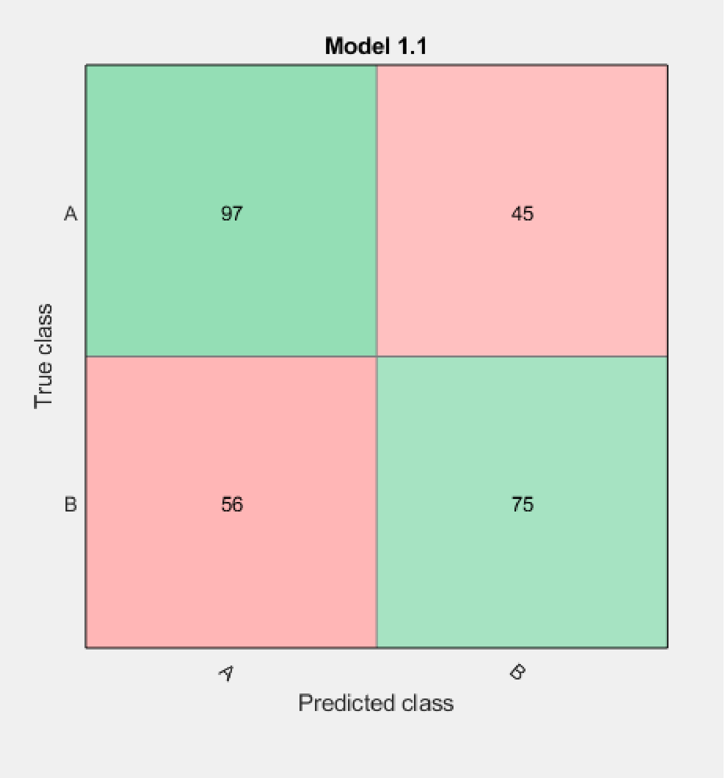
Subspace KNN



True class \ Predicted class	J	K	L	M	N	O	P	Q	R	S
J	2			1	1					
K	1	2	2							
L	1	2			1					
M	4	3	6	13	8	1	5	7	3	8
N	5	1	7	10	4	7	4	5	7	5
O	5		5	6	5	4	3	8	4	5
P	2	2	1	7	4	4	1	5	9	12
Q	4			2	5	4	3	12	1	7
R	1		1		1	3	1	2	2	4
S										2

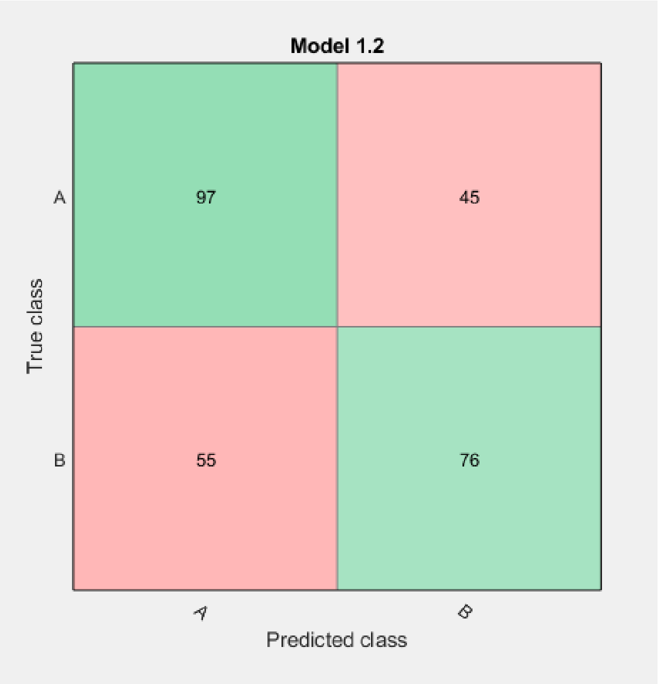
RUSBoosted Trees

3.4 Median Bin



True class \ Predicted class	A	B
A	97	45
B	56	75

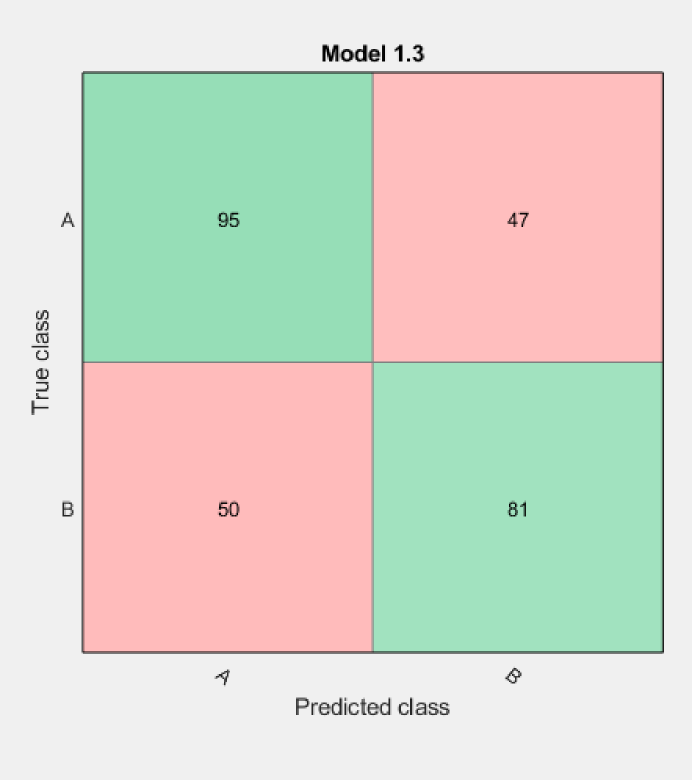
Fine Tree



Model 1.2

True class \ Predicted class	A	B
A	97	45
B	55	76

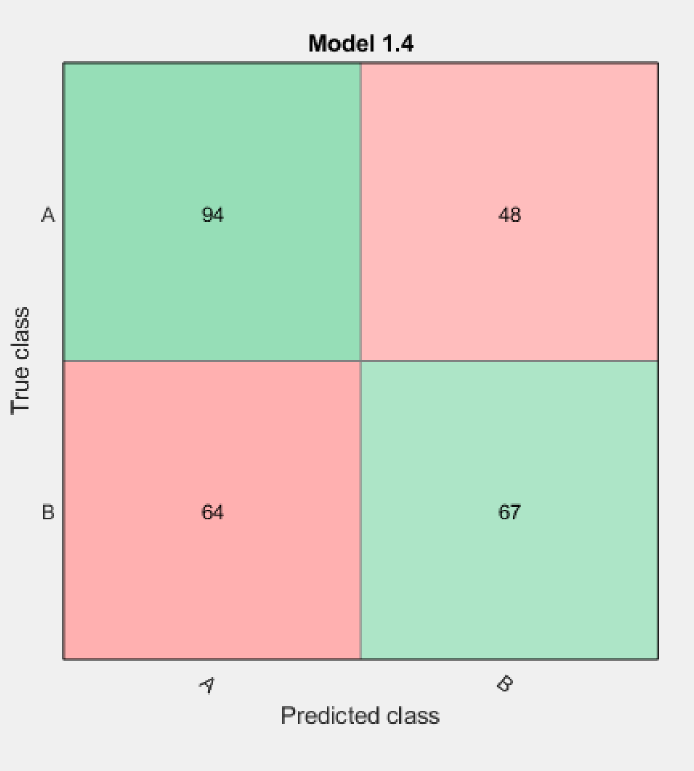
Medium Tree



Model 1.3

True class \ Predicted class	A	B
A	95	47
B	50	81

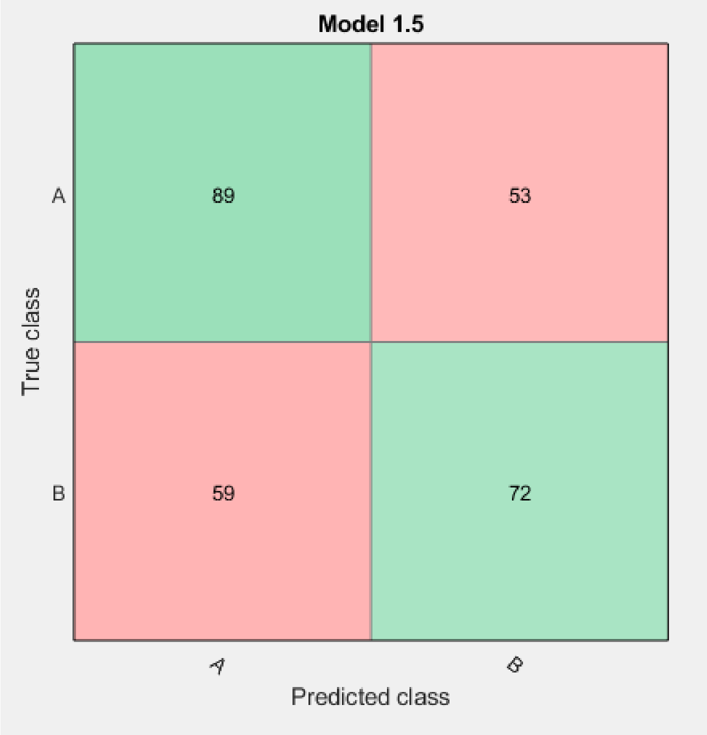
Coarse Tree



Model 1.4

	A	B
A	94	48
B	64	67

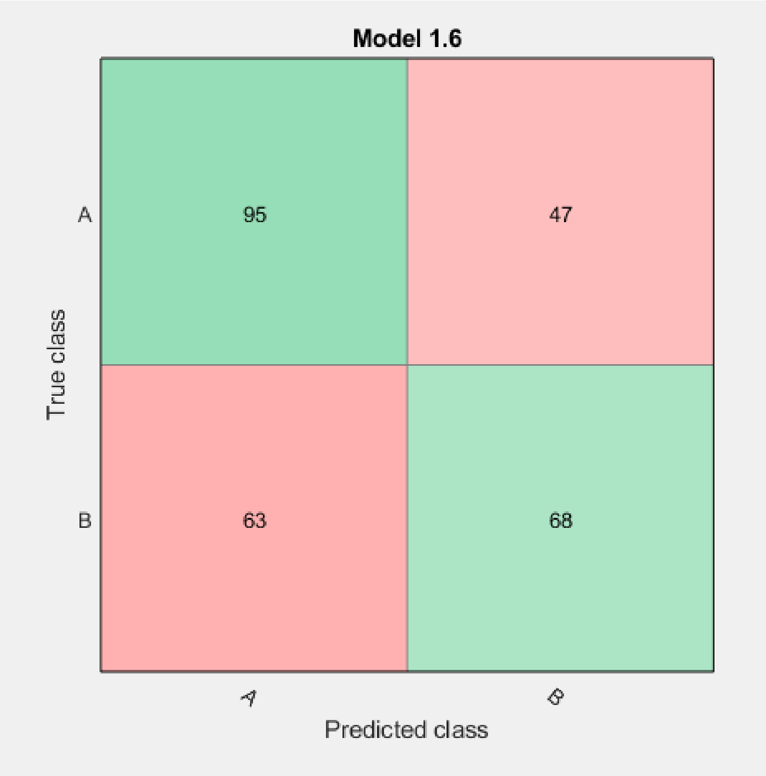
Linear Discriminant



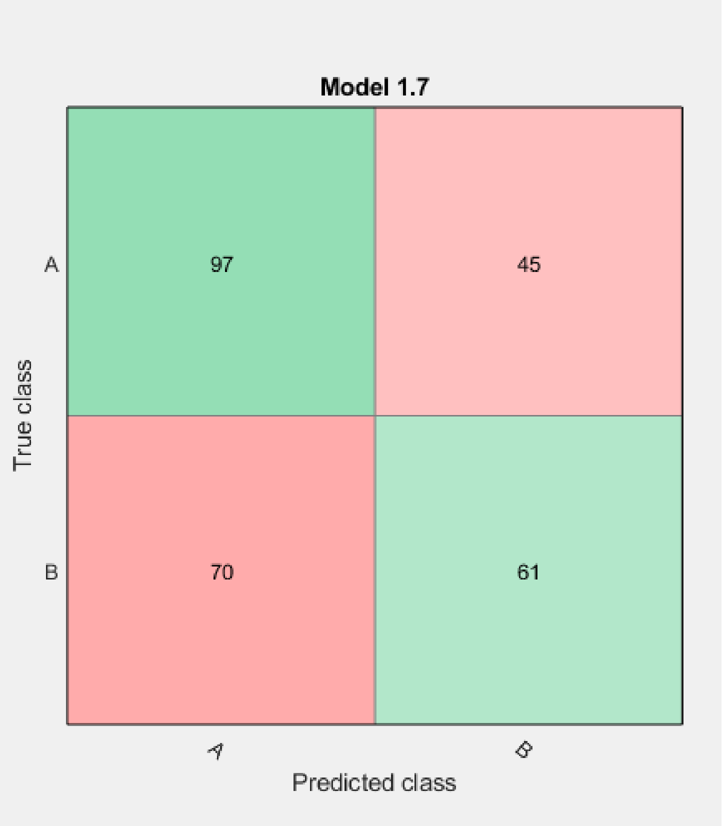
Model 1.5

	A	B
A	89	53
B	59	72

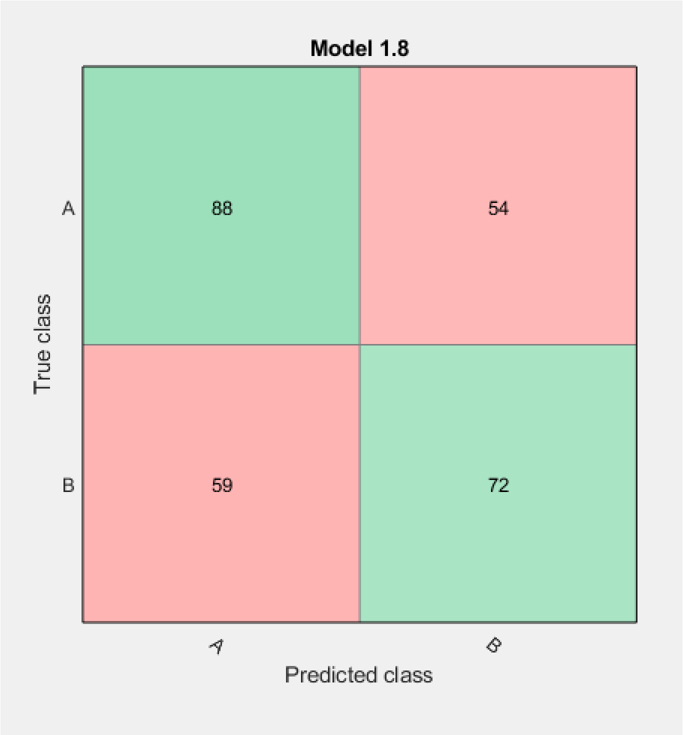
Quadratic Discriminant



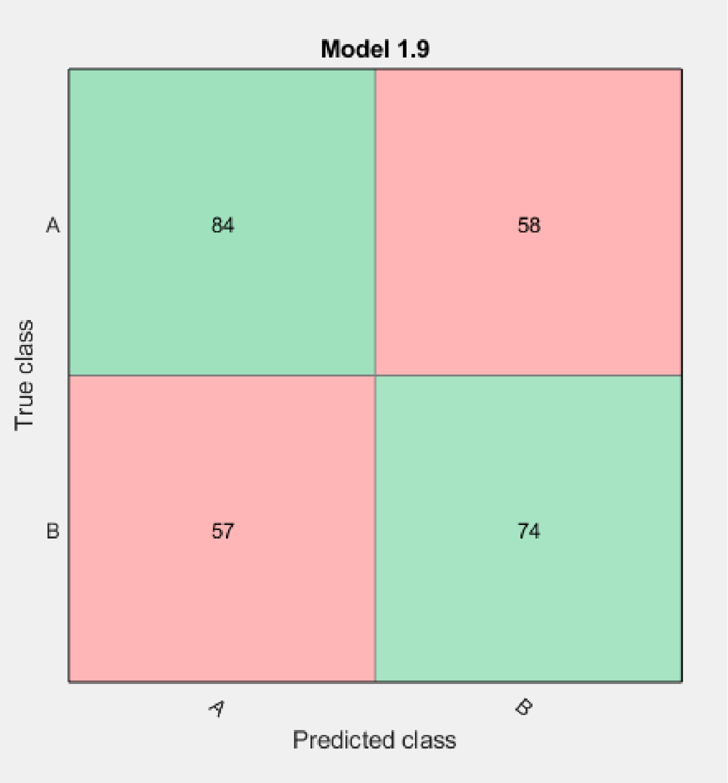
Logistic Regression



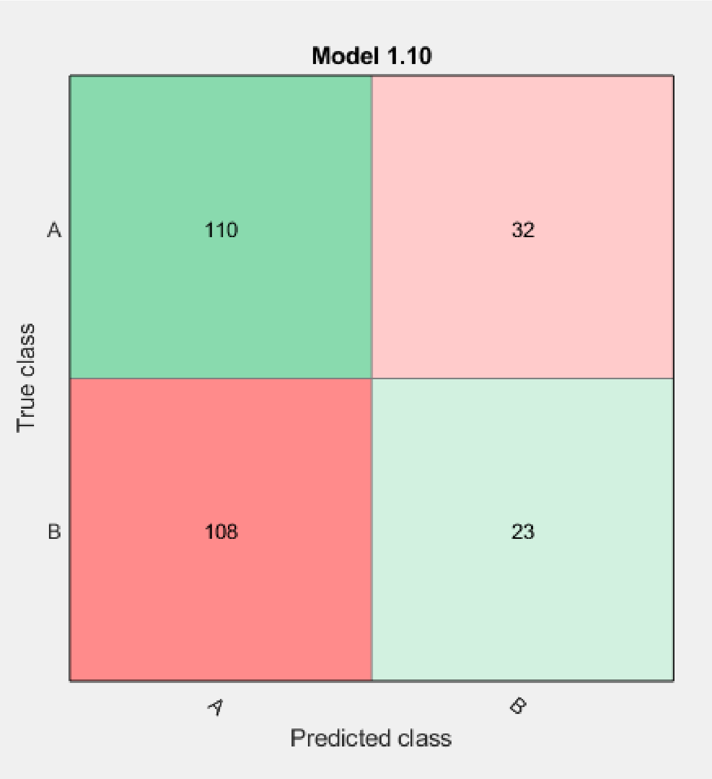
Linear SVM



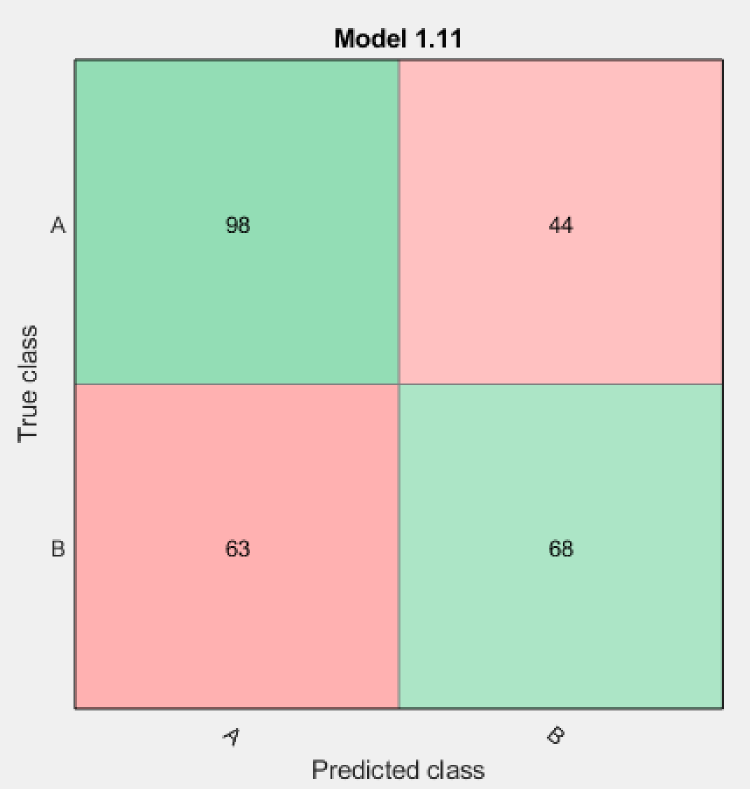
Quadratic SVM



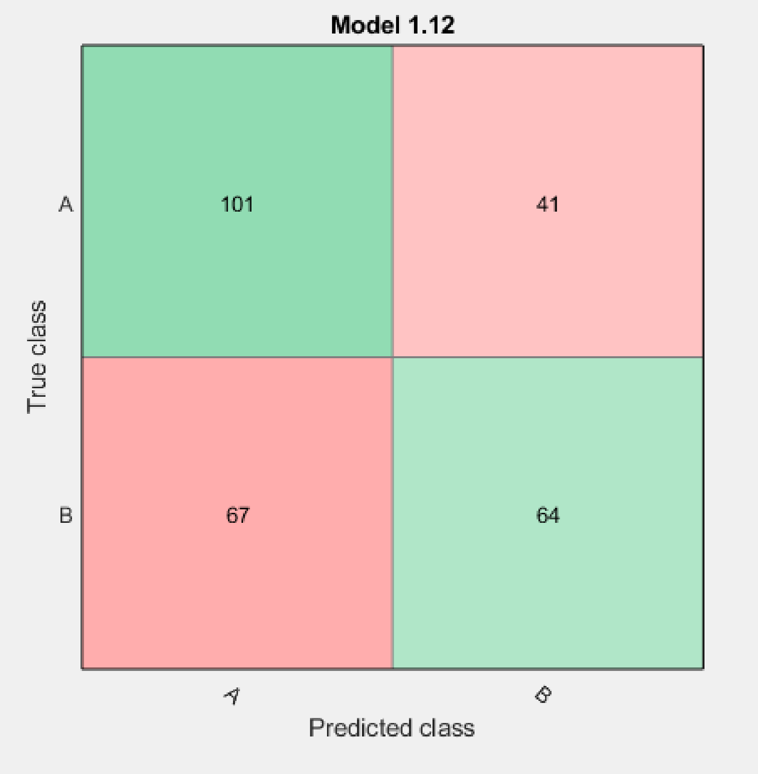
Cubic SVM



Fine Gaussian SVM

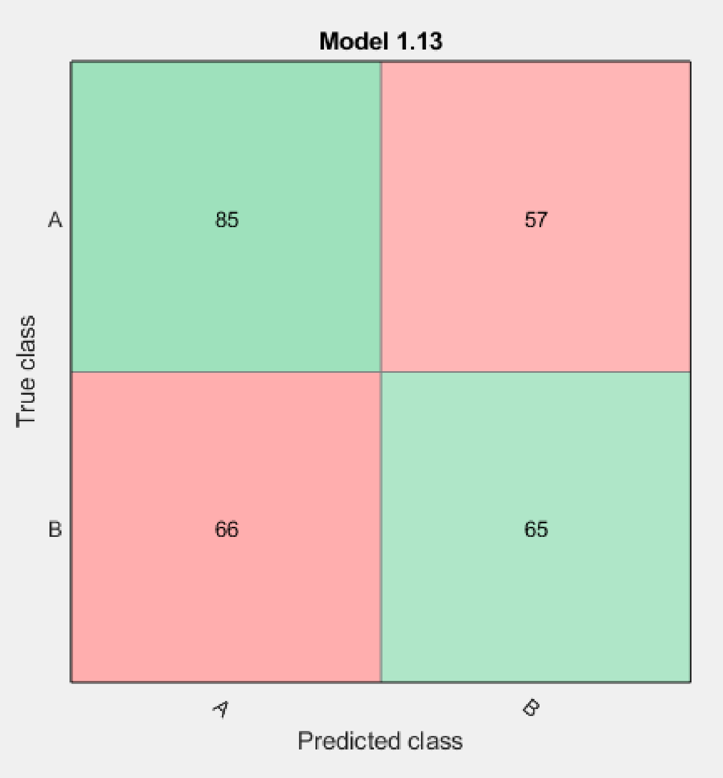


Medium Gaussian SVM



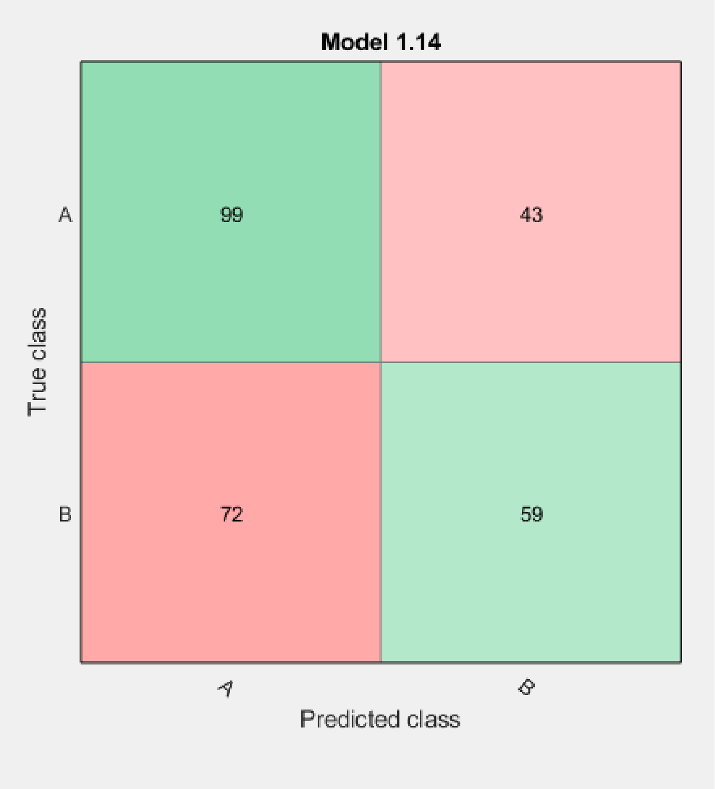
	A	B
A	101	41
B	67	64

Coarse Gaussian SVM

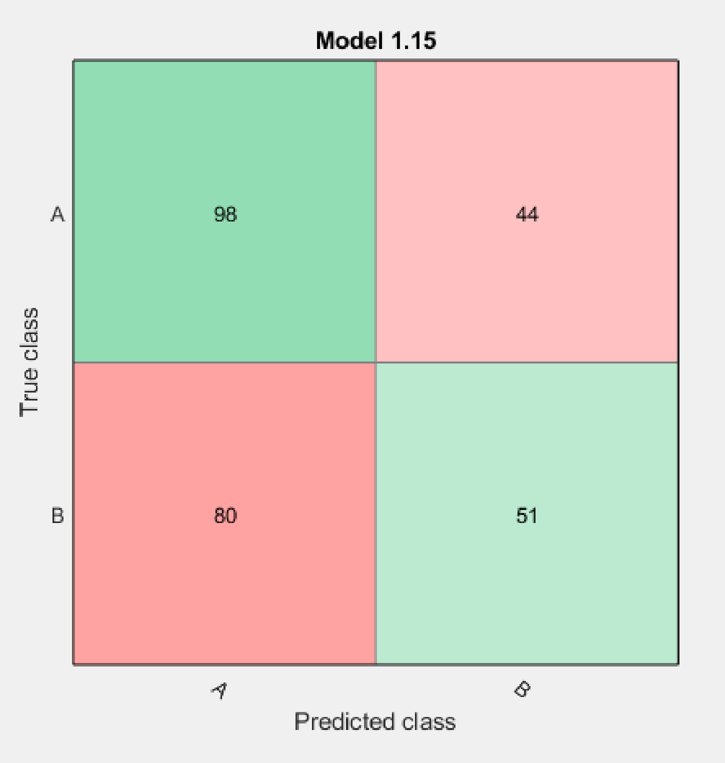


	A	B
A	85	57
B	66	65

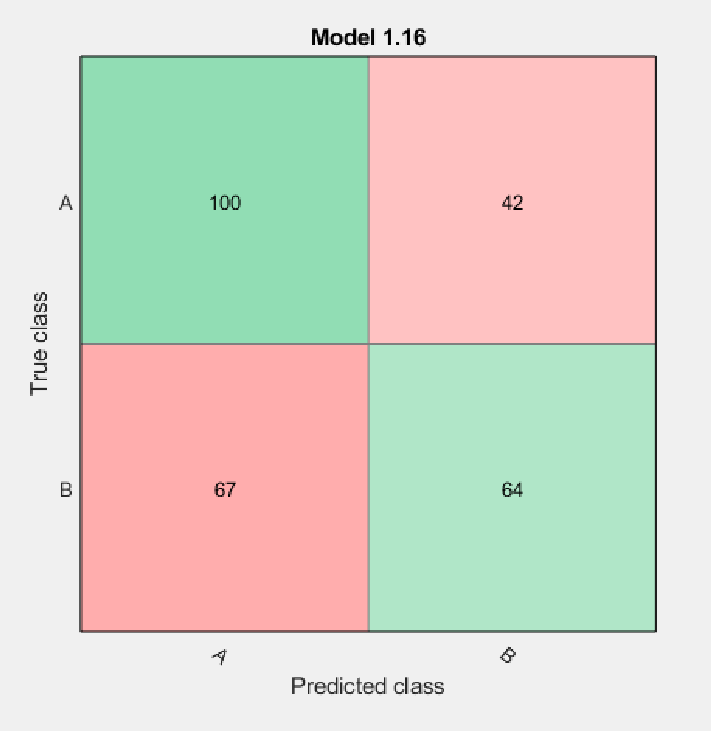
Fine KNN



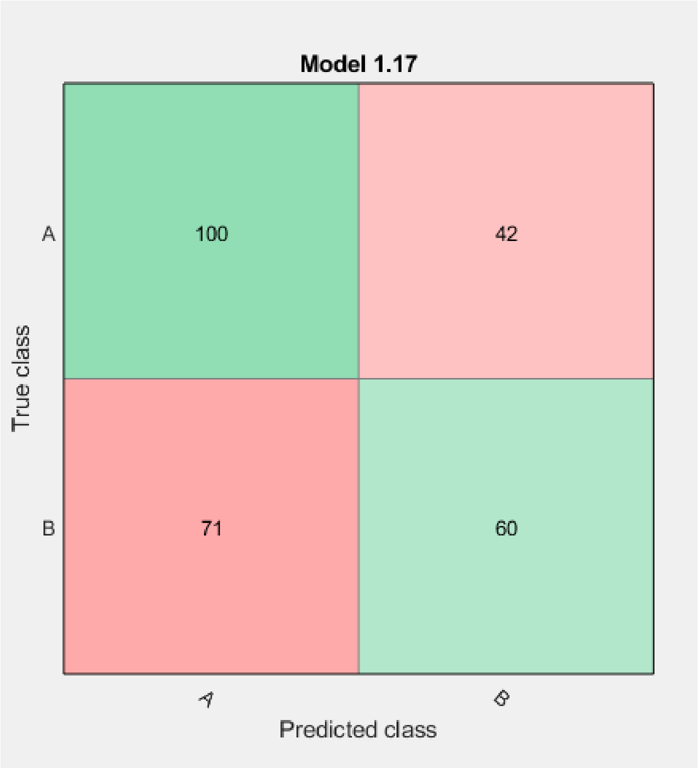
Medium KNN



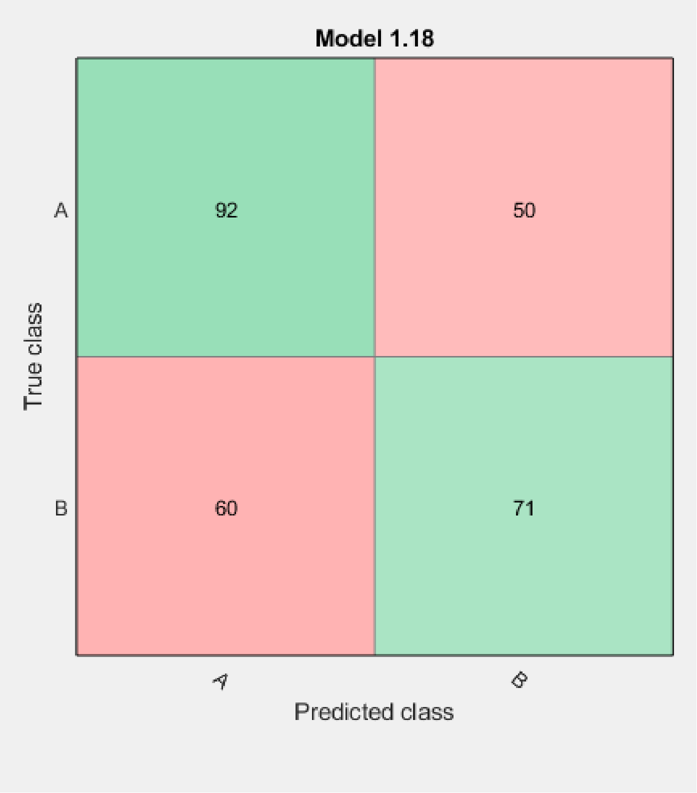
Coarse KNN



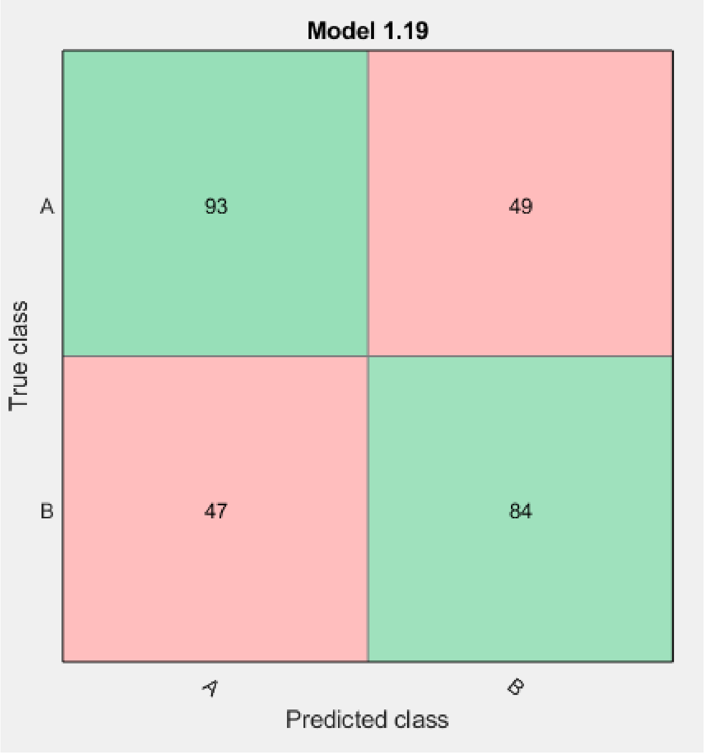
Cosine KNN



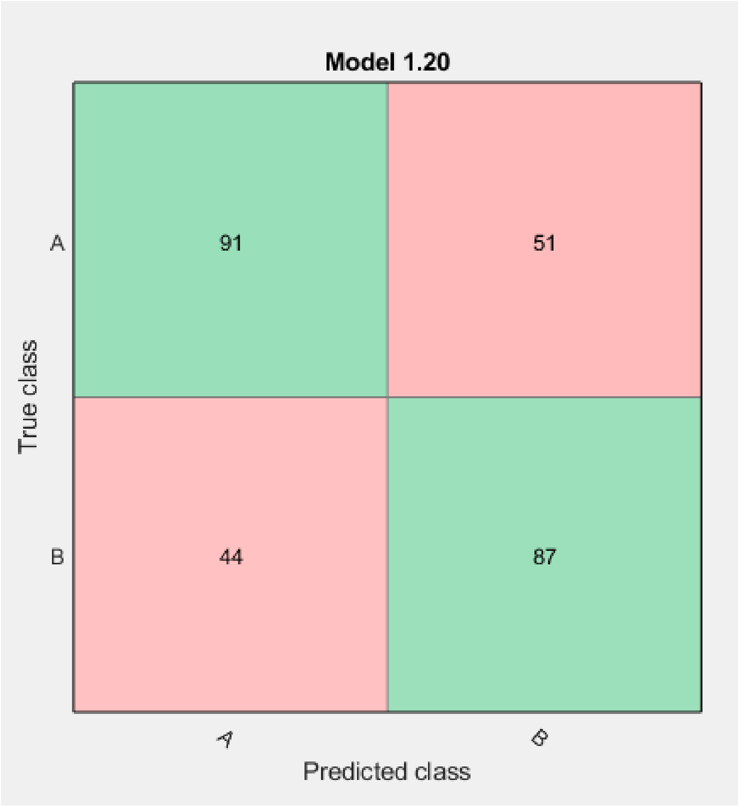
Cubic KNN



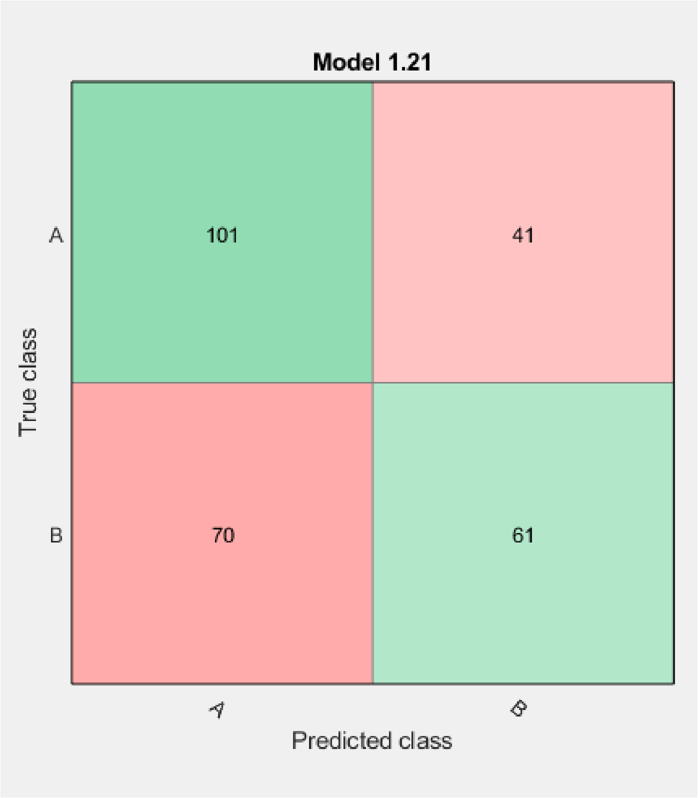
Weighted KNN



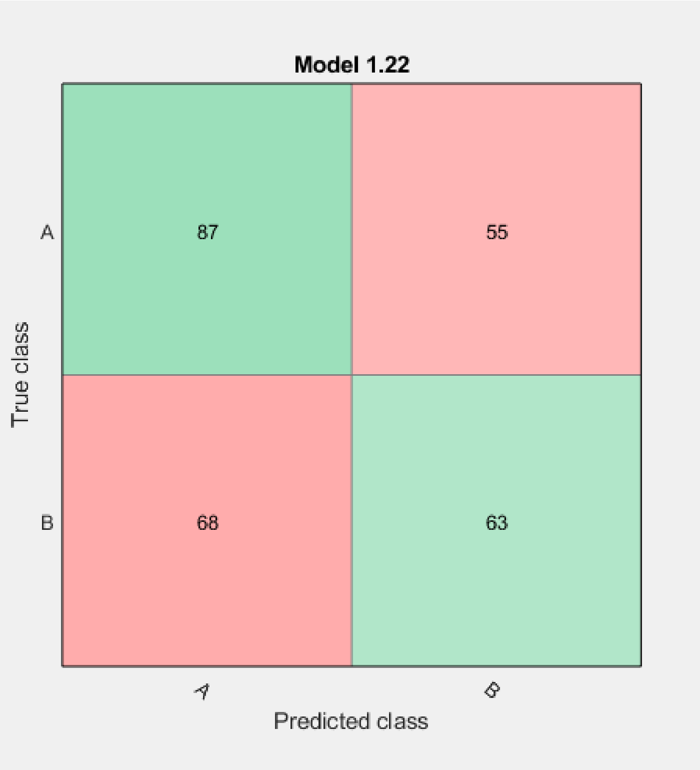
Boosted Trees



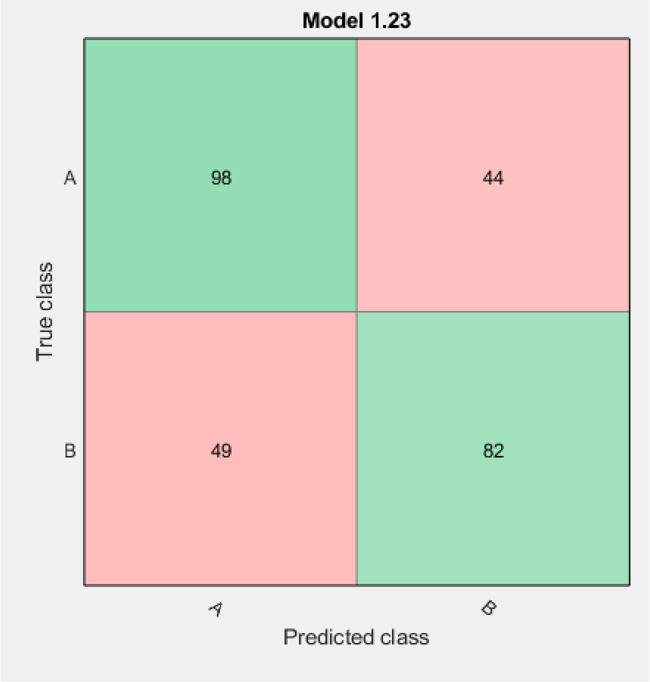
Bagged Trees



Subspace Discriminant

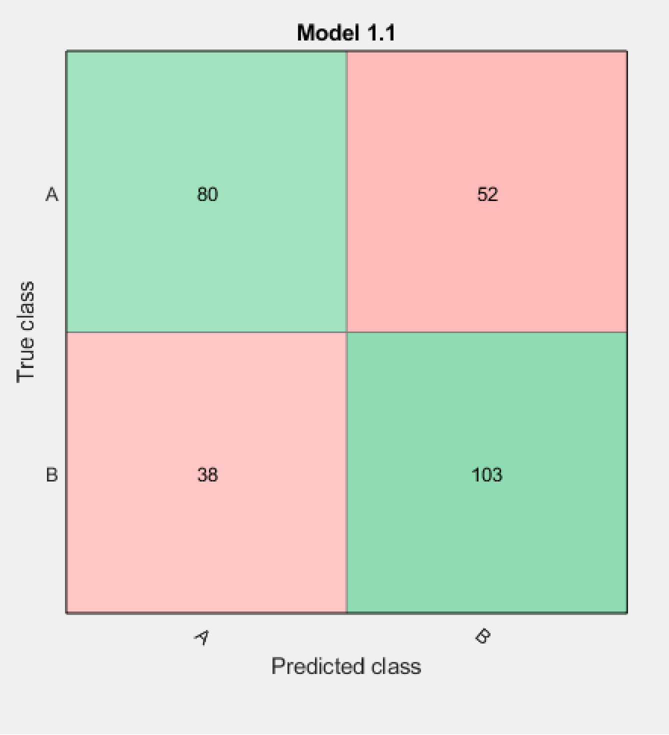


Subspace KNN

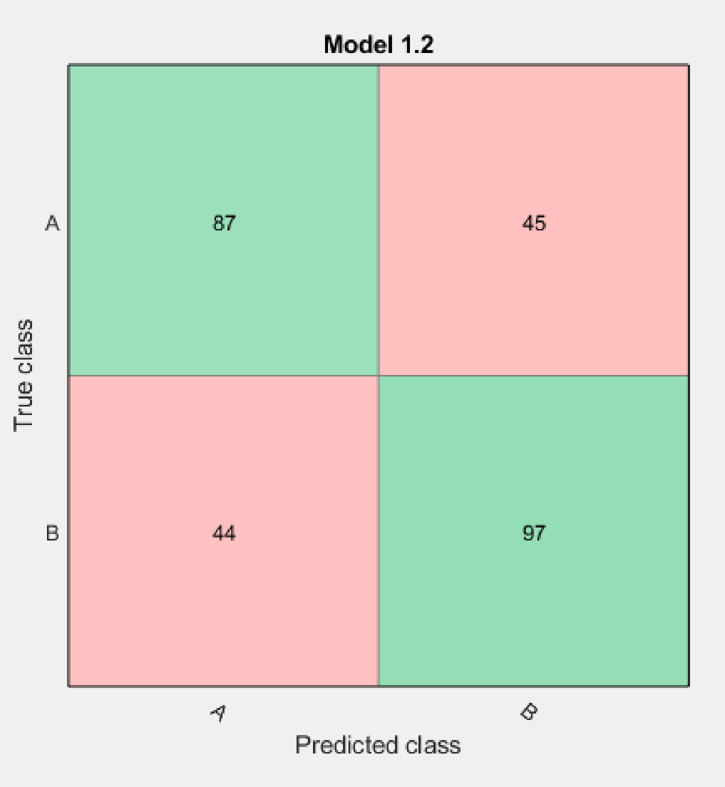


RUSBoosted Trees

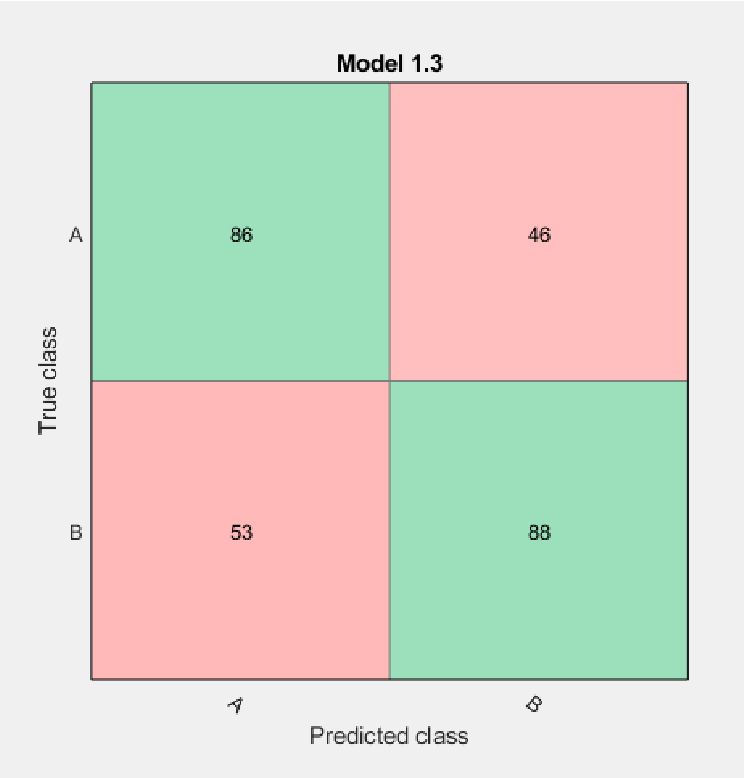
3.5 Mean Bin



Fine Tree

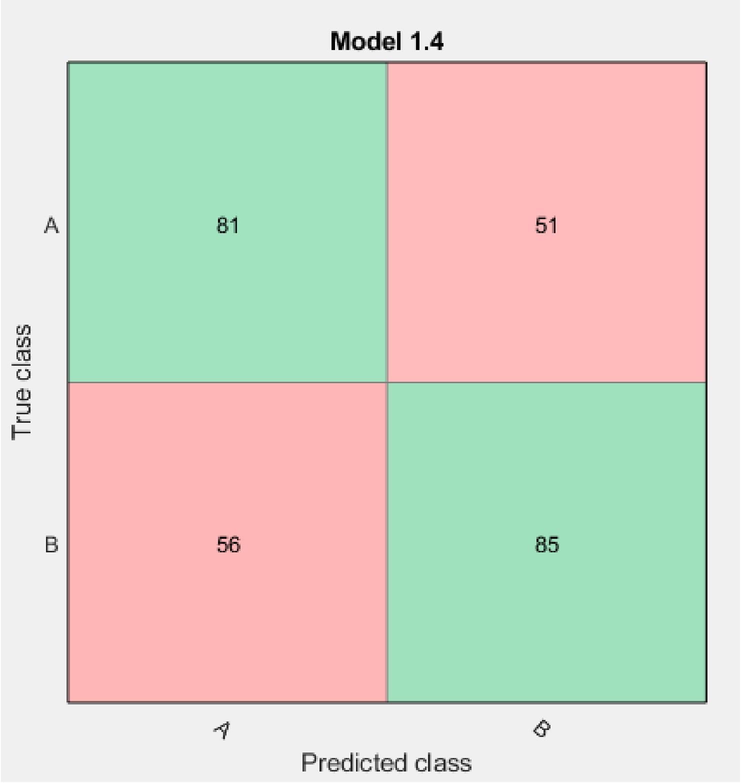


Medium Tree



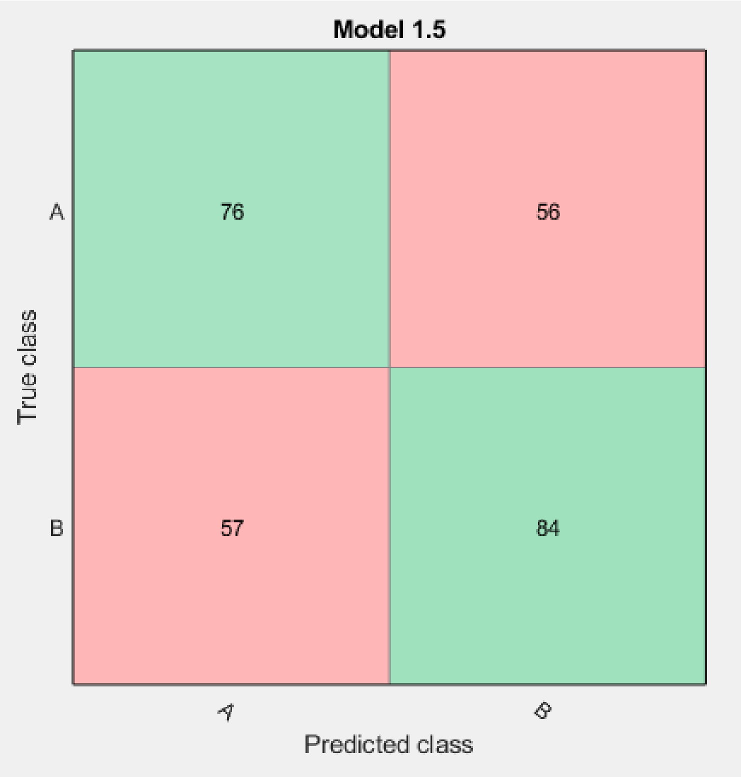
	A	B
A	86	46
B	53	88

Coarse Tree

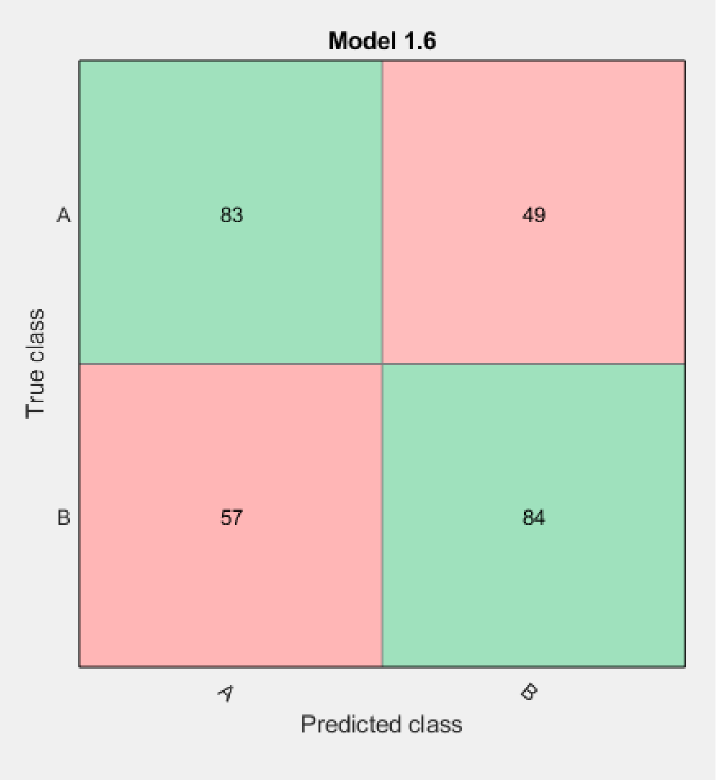


	A	B
A	81	51
B	56	85

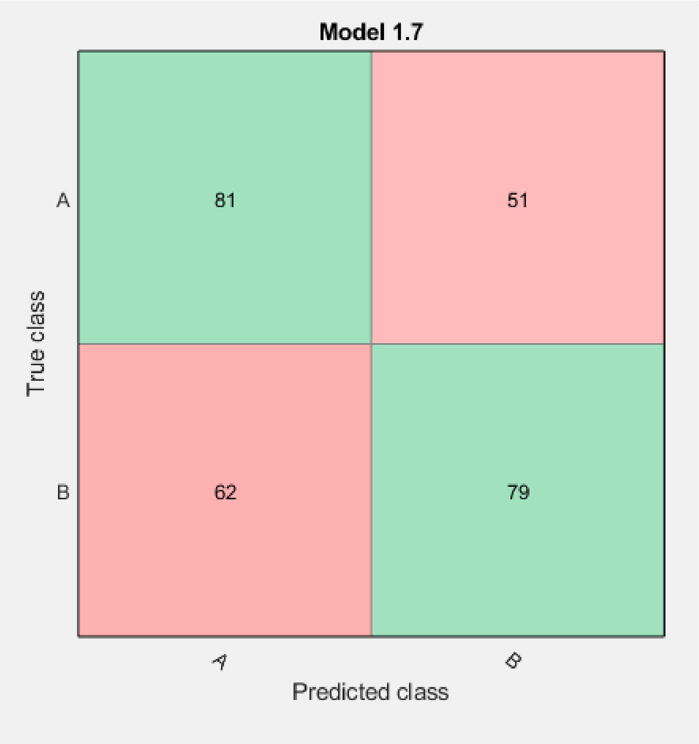
Linear Discriminant



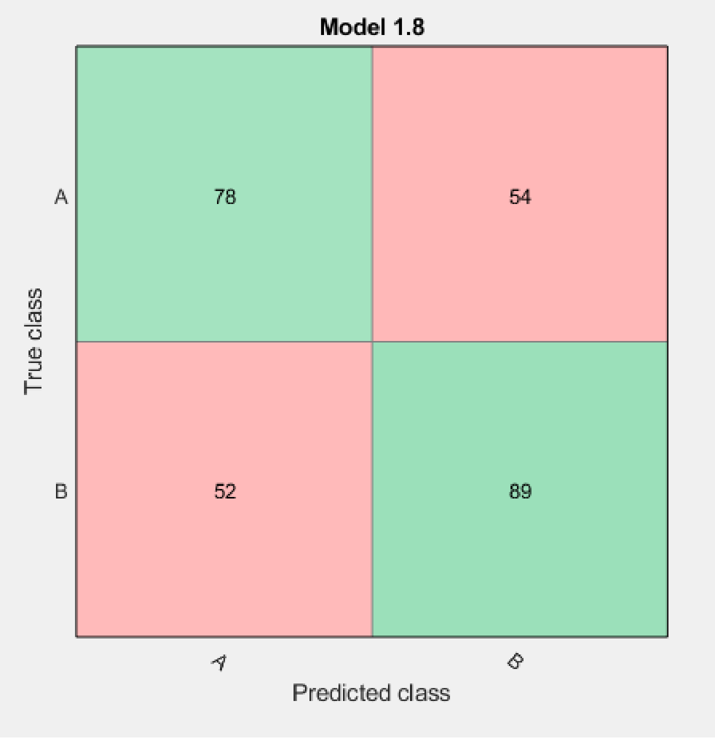
Quadratic Discriminant



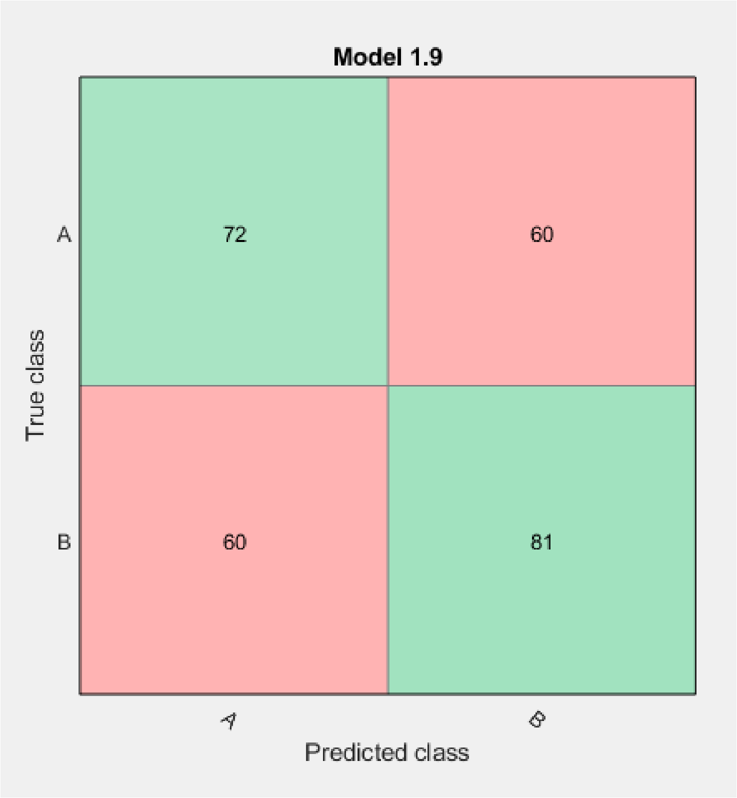
Logistic Regression



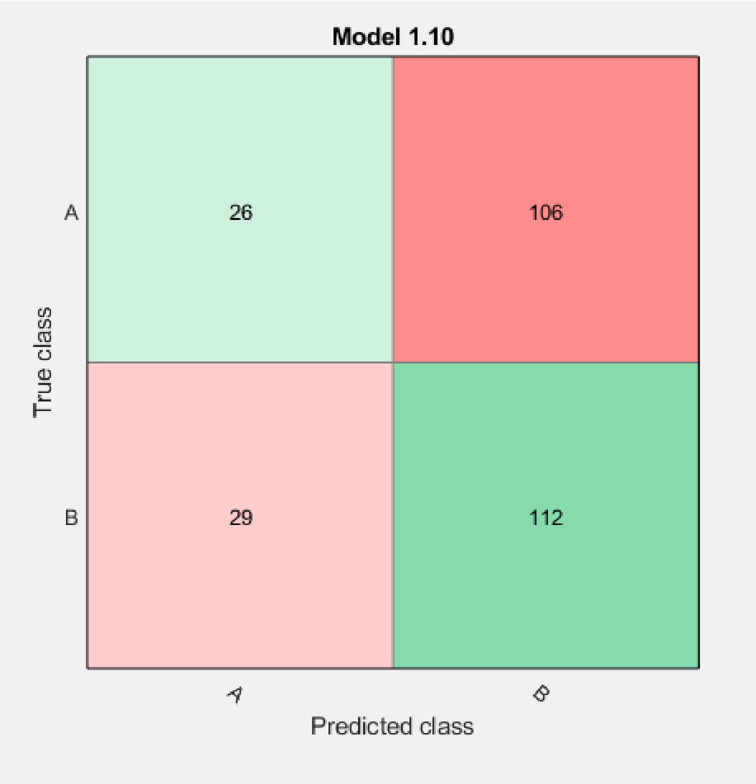
Linear SVM



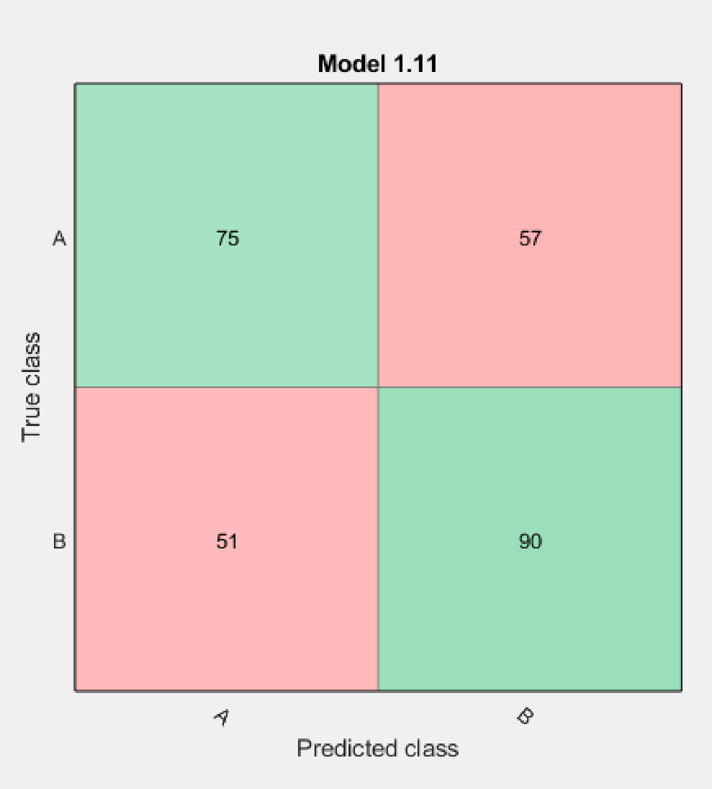
Quadratic SVM



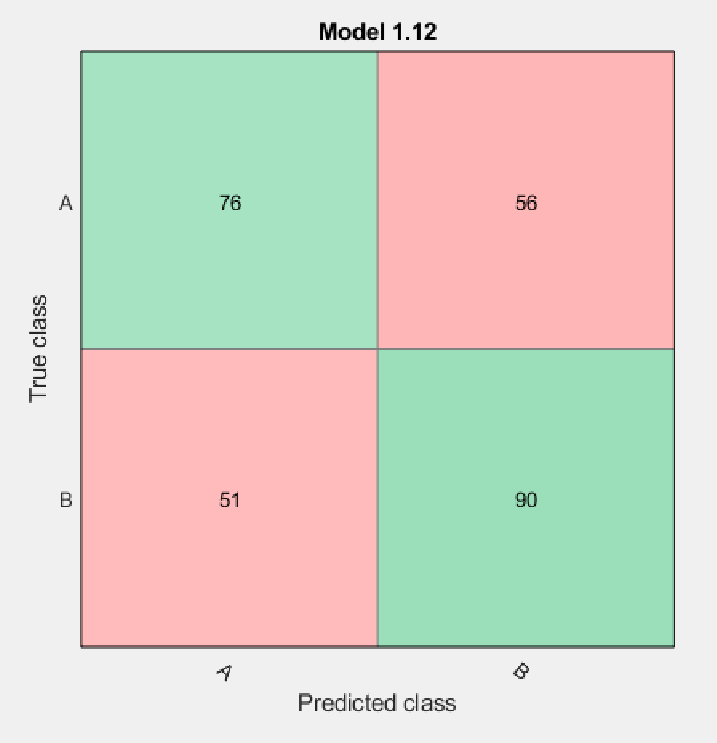
Cubic SVM



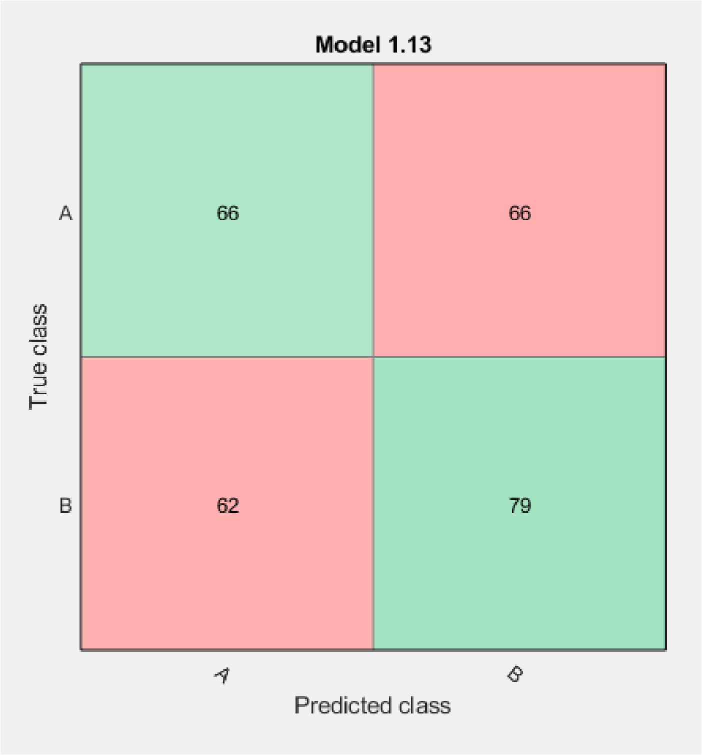
Fine Gaussian SVM



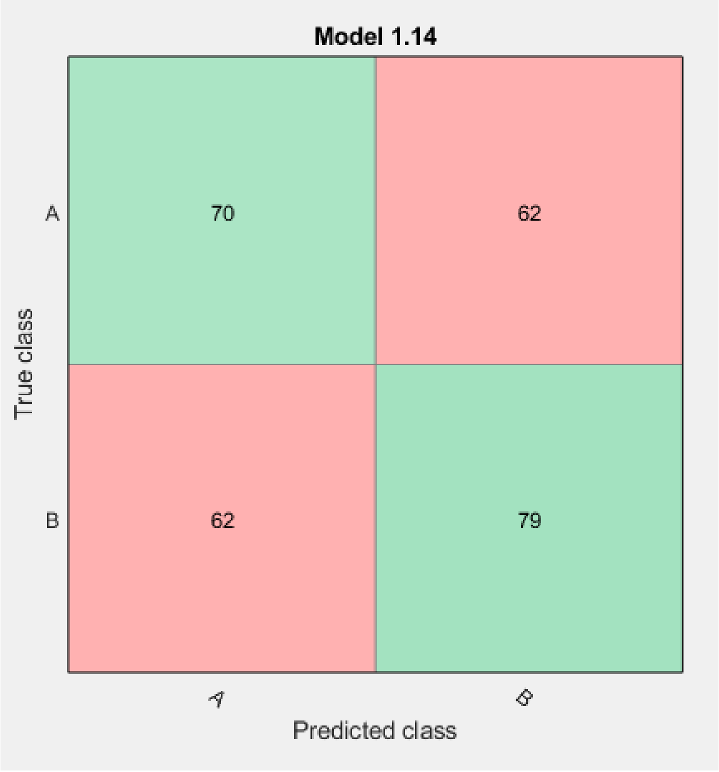
Medium Gaussian SVM



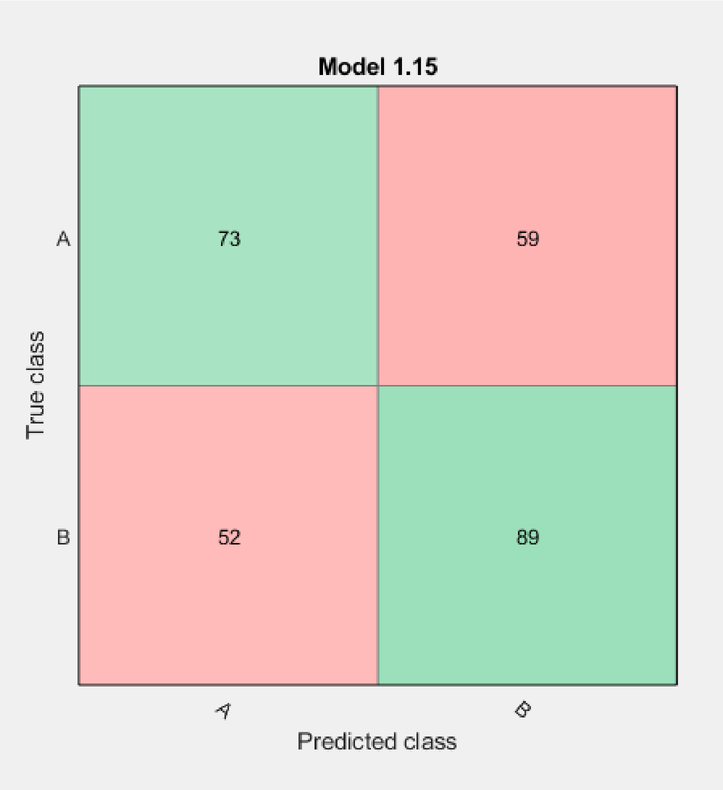
Coarse Gaussian SVM



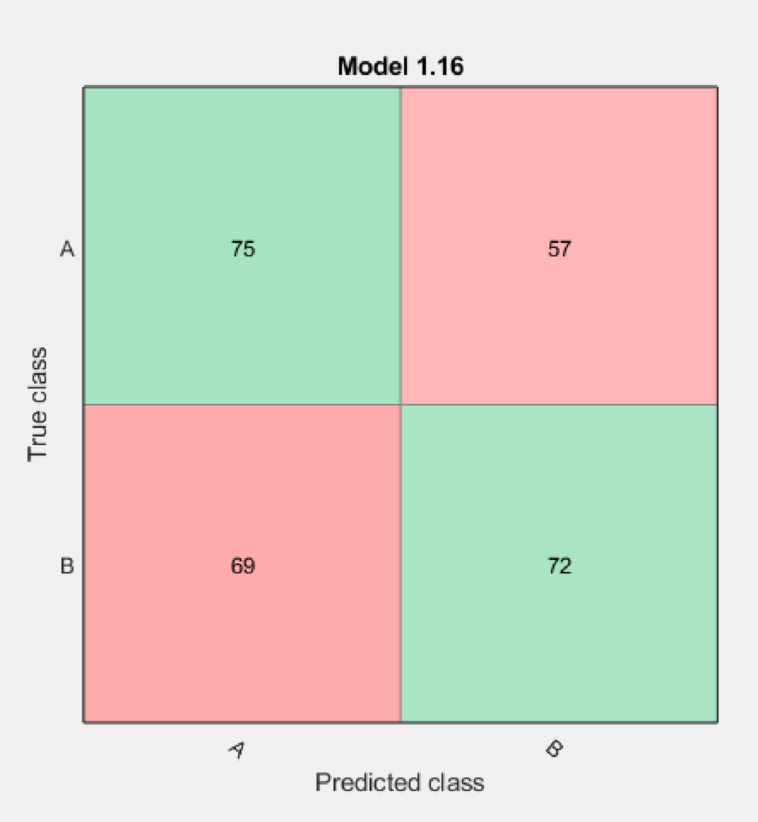
Fine KNN



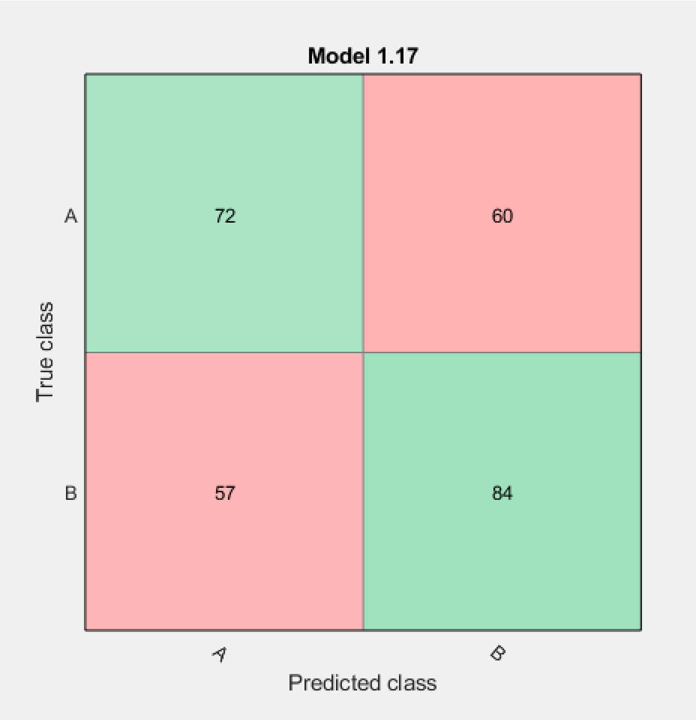
Medium KNN



Coarse KNN

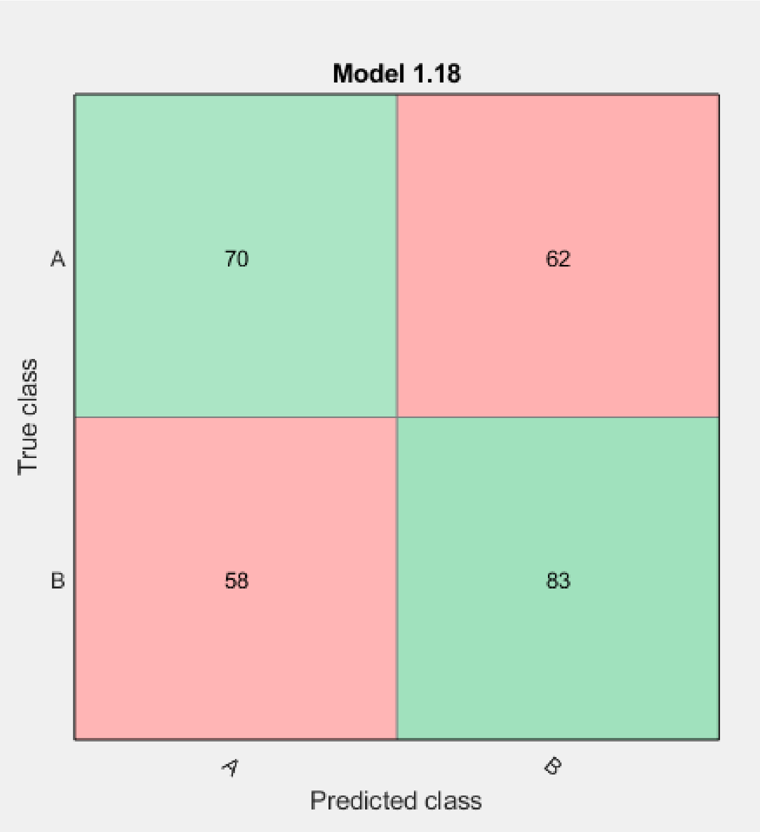


Cosine KNN



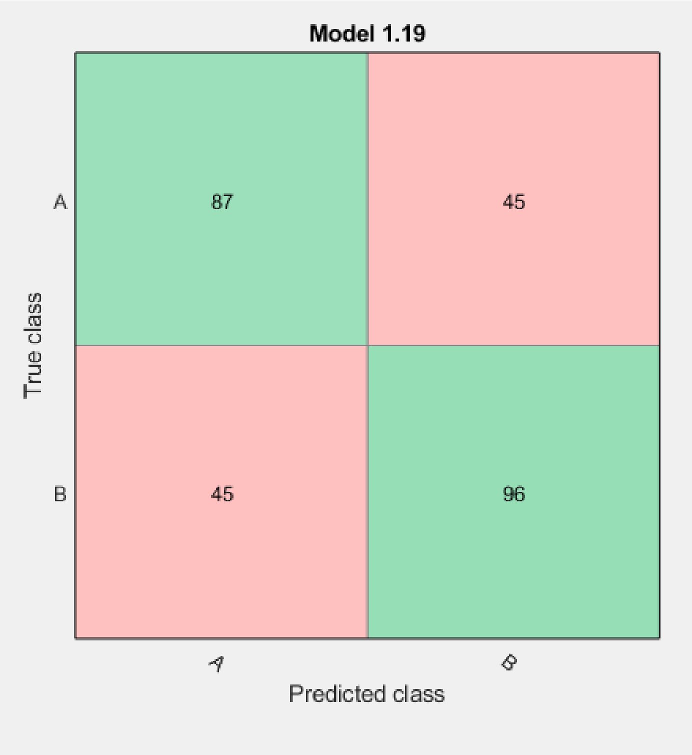
	A	B
A	72	60
B	57	84

Cubic KNN

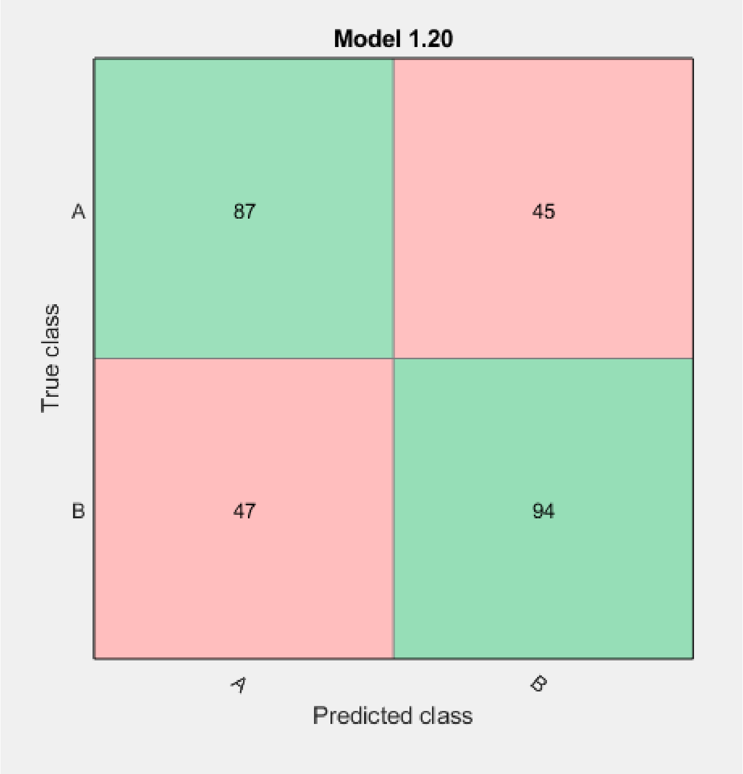


	A	B
A	70	62
B	58	83

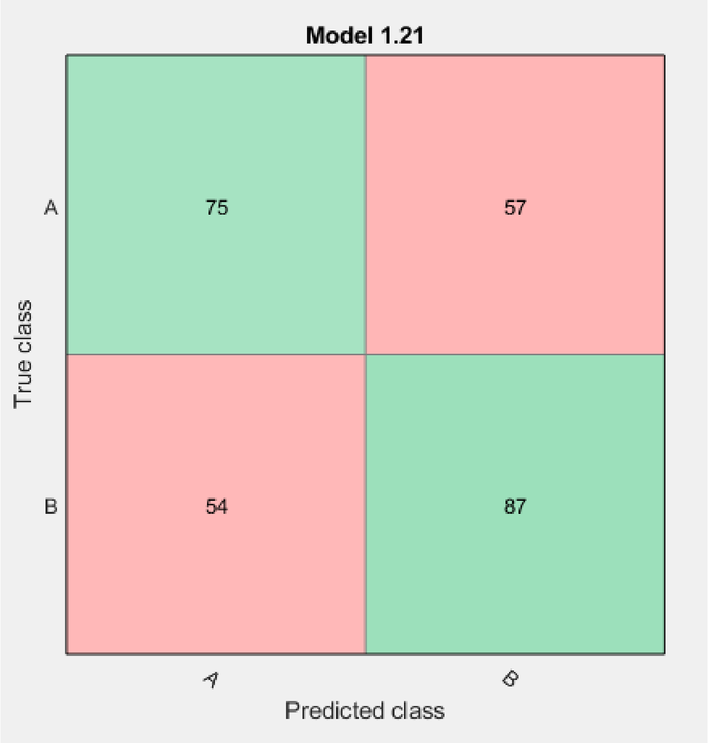
Weighted KNN



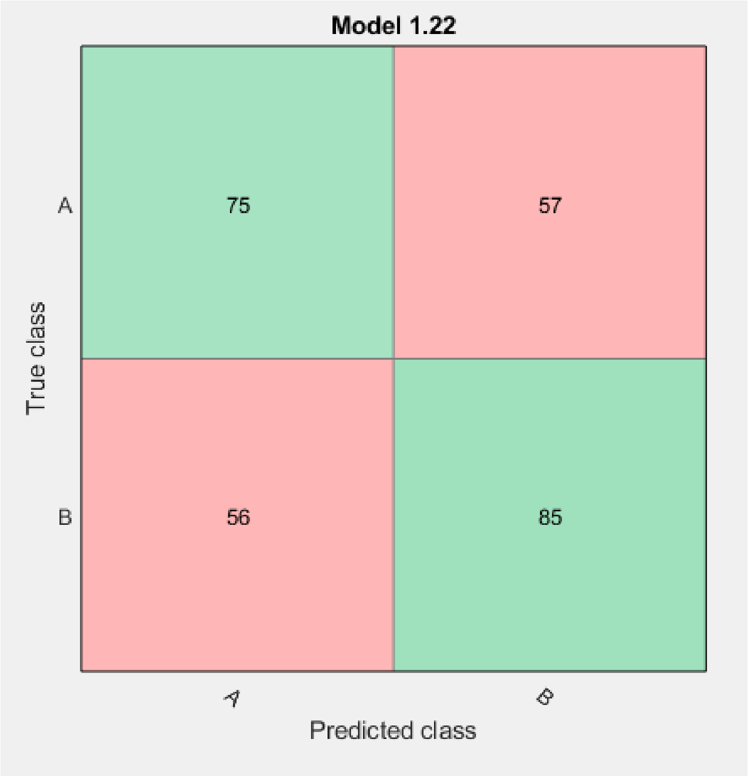
Boosted Trees



Bagged Trees

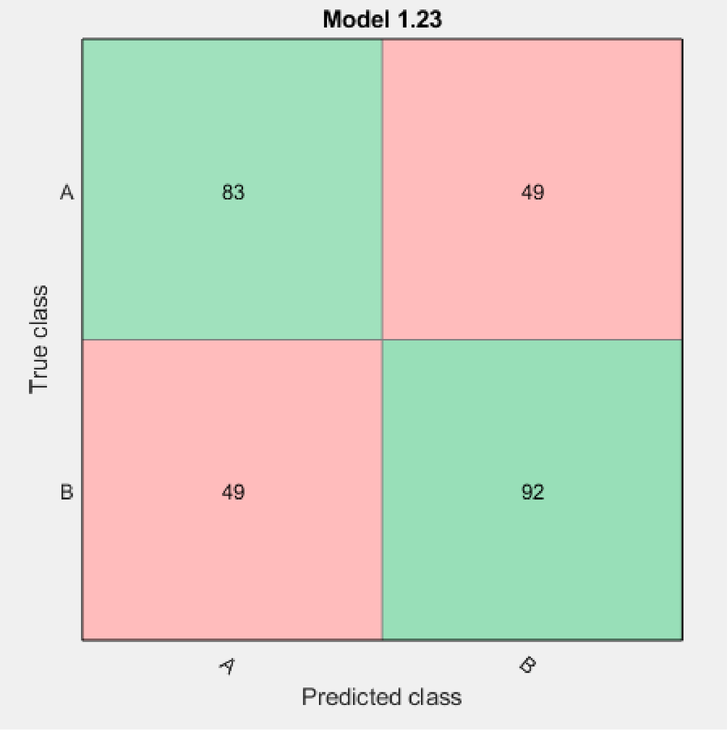


Subspace Discriminant



True class \ Predicted class	A	B
A	75	57
B	56	85

Subspace KNN



True class \ Predicted class	A	B
A	83	49
B	49	92

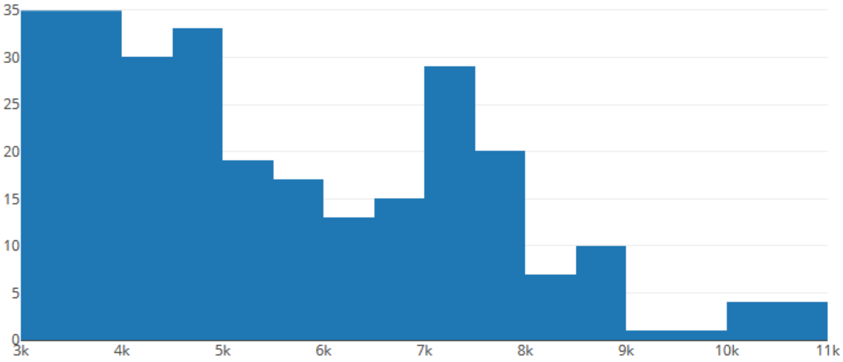
RUSBoosted Trees

4 Feature Distribution

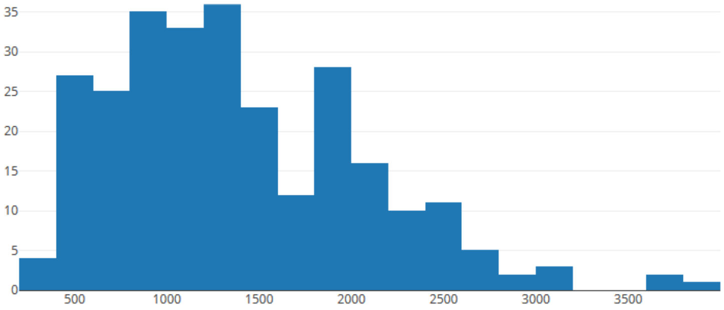
4.1 Descriptive Table

Feature	Mean	Median	Standard Deviation
EEQ	72.0476190476	72	10.3042330915
Daily Total Steps	5595.2124542124	5181	1881.3048309487
Playing Steps	1389.2527472527	1265	679.7110826660
Distance Traveled	0.2064761904	0.016	0.2978601098
Distance Traveled From Home	0.3242747252	0.31	0.1129791448
Temperature	46.2868131868	41	12.2171189159
Precipitation	0.0732600732	0	0.3757034843
Wind	7.7271062271	8.1	5.8683666203
Humidity	55.4835164835	60	12.0213753188
Times App Opened	13.8095238095	14	3.0503078797
Time Played	72.9871794871	73.14	12.2478997419
Average Session Length	19.6520146520	20	2.9895675701
Earliest Open	10.2733808840	10.29820944	0.9496682880
Latest Close	19.2417529170	19.25568834	1.1461834431
Average Lag Time	178.401828302	194.018282931	27.819283904

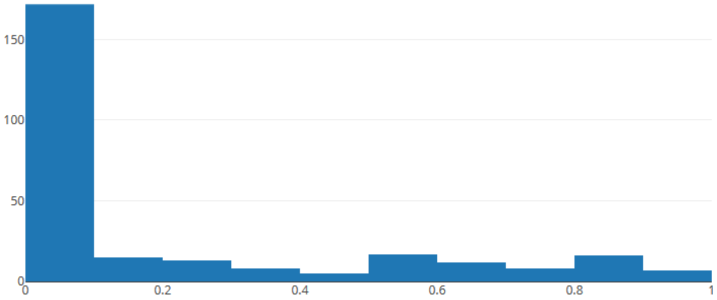
4.2 Feature Histograms



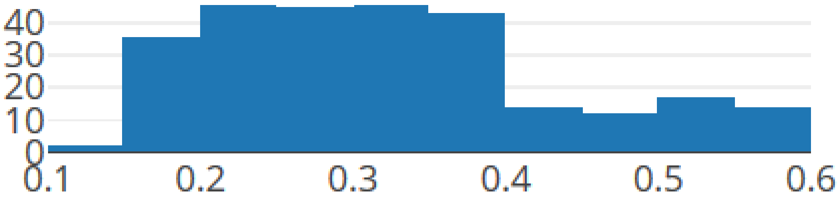
Daily Total Steps



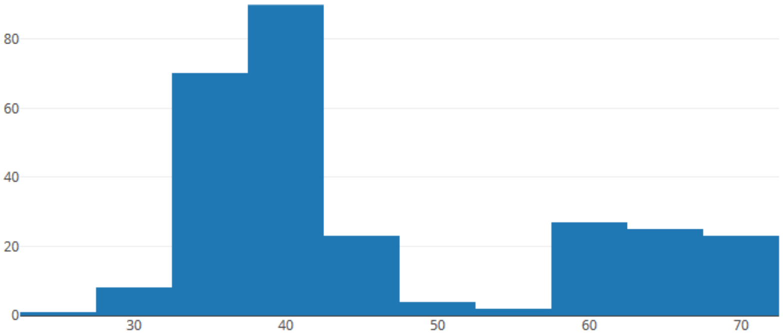
Playing Steps



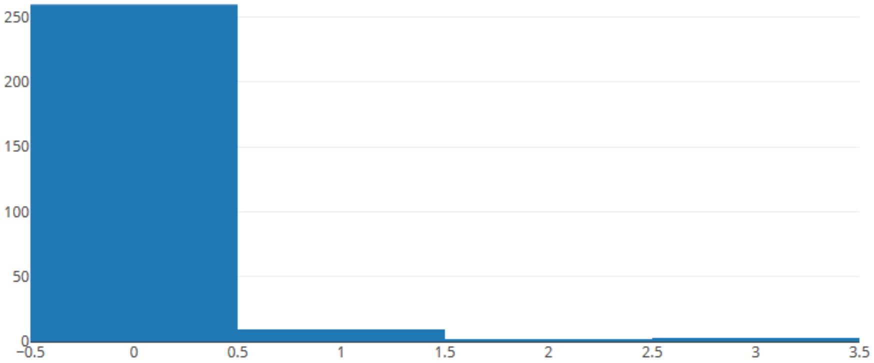
Distance Traveled While Playing



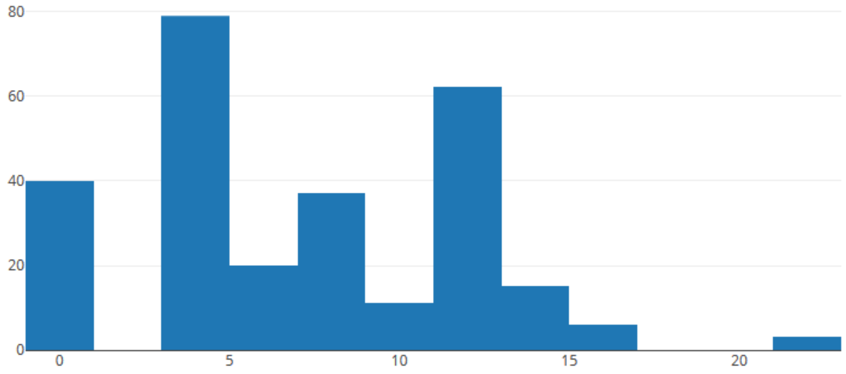
Max Distance Traveled From Home While Playing



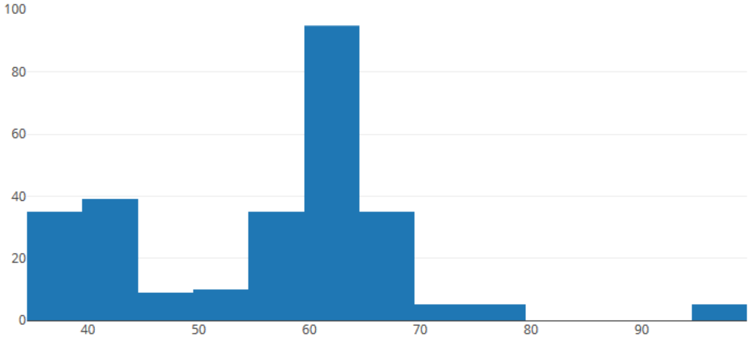
Average Daily Temperature



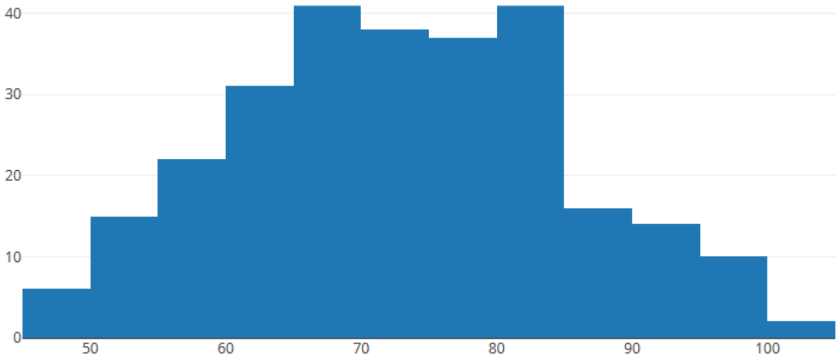
Precipitation



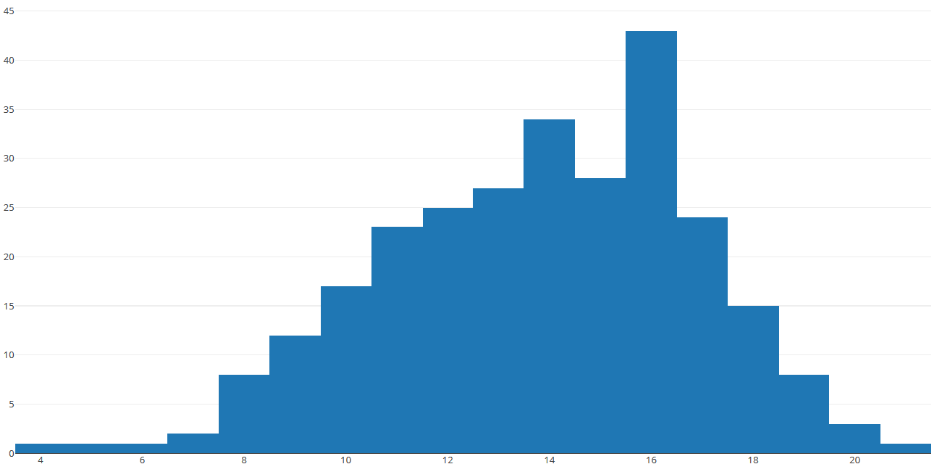
Wind Speed



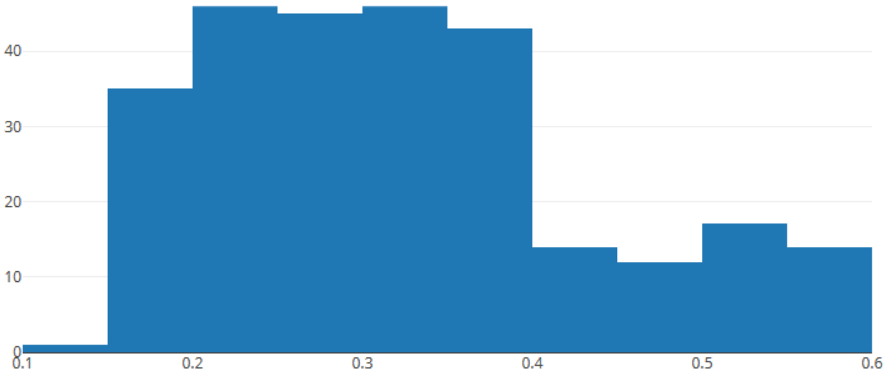
Humidity



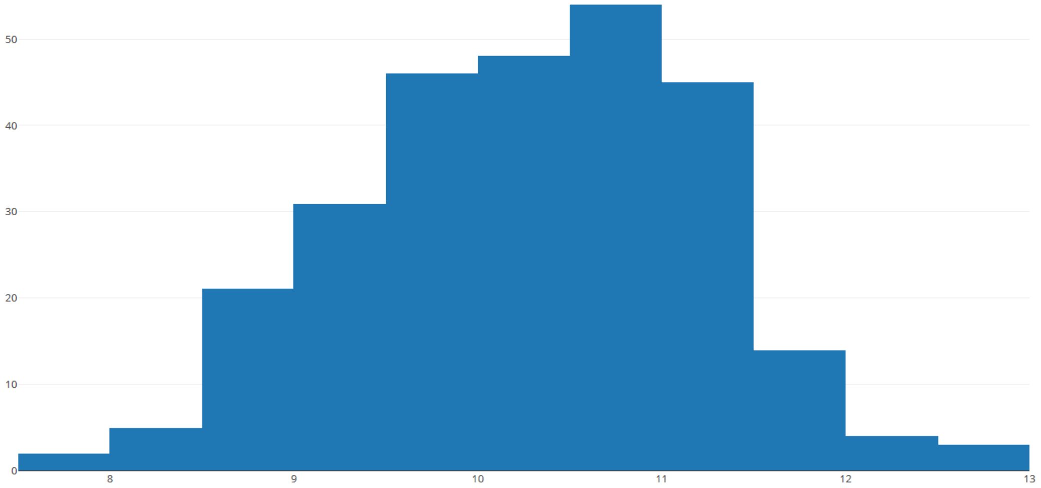
Time Played



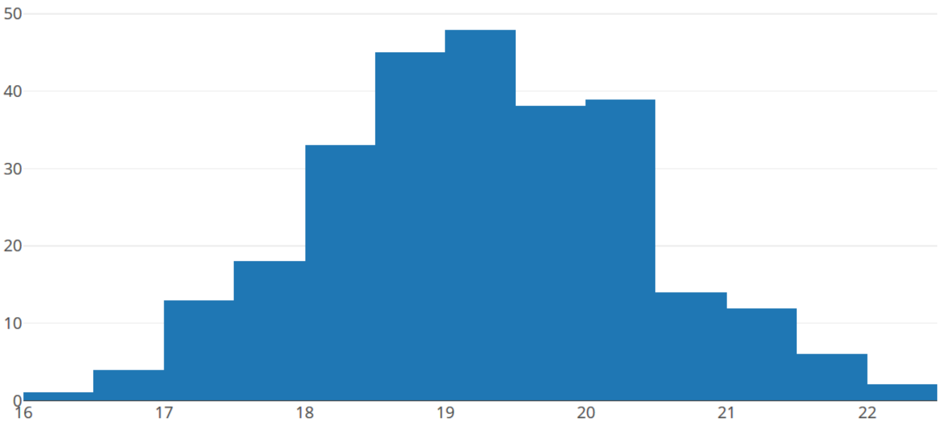
Times App Opened



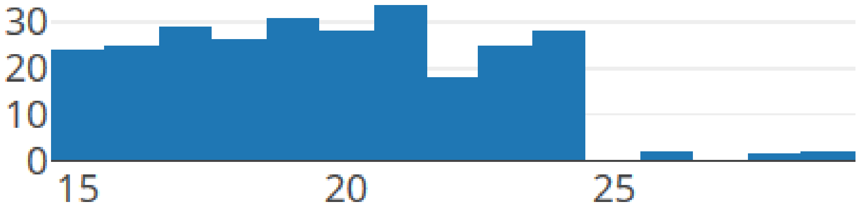
Max Distance Traveled From Home



Earliest Open

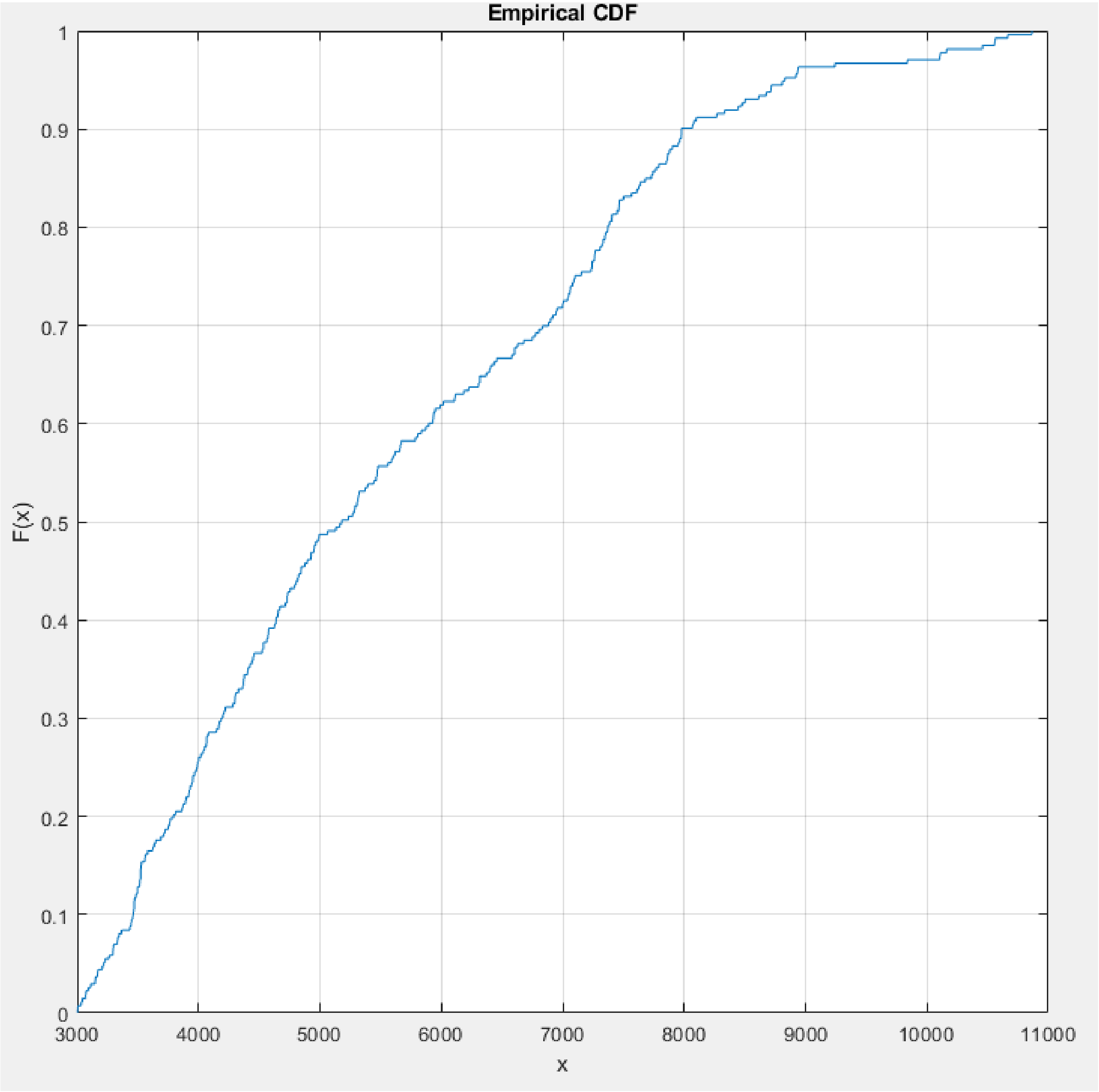


Latest Close

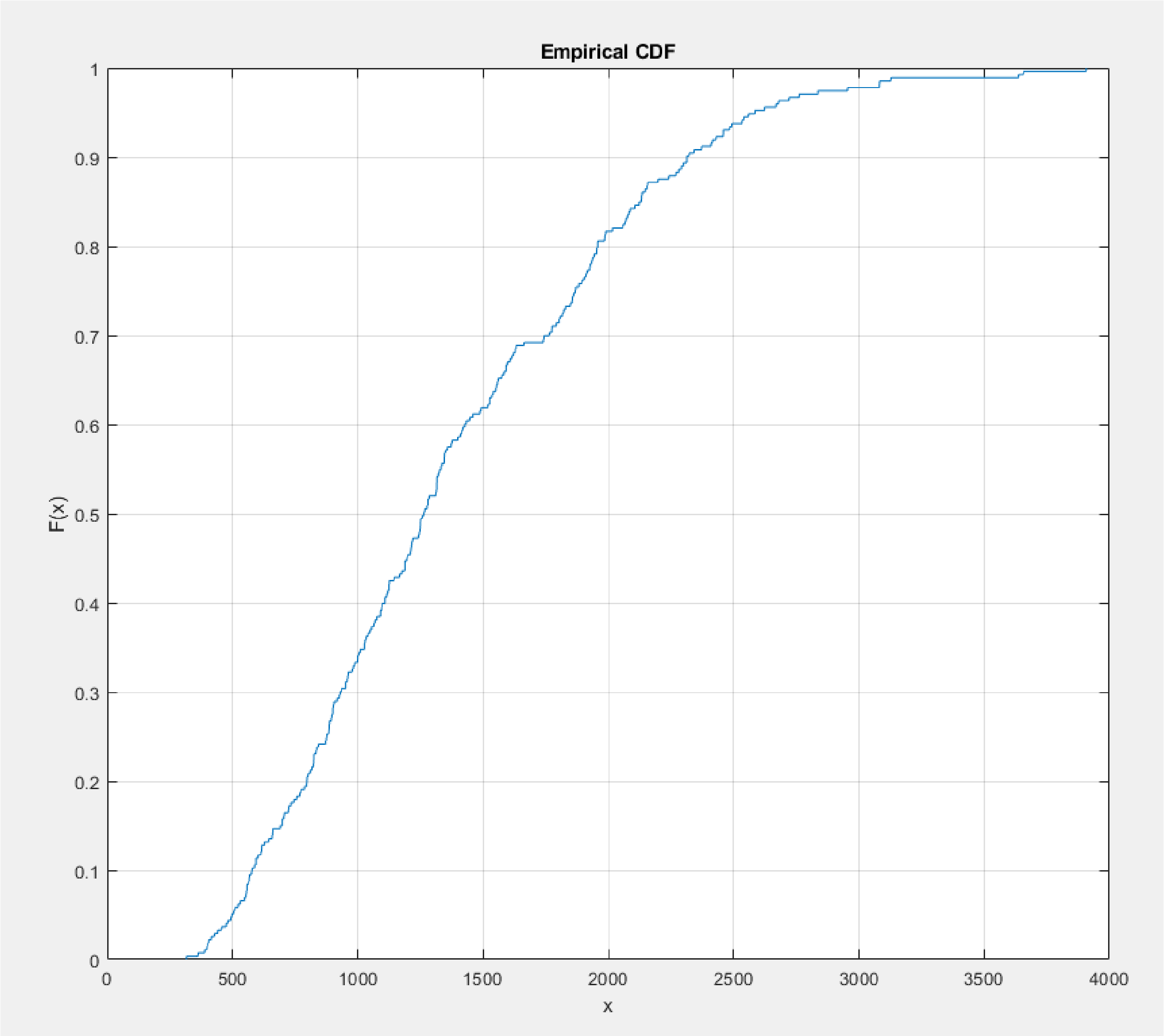


Average Session Length

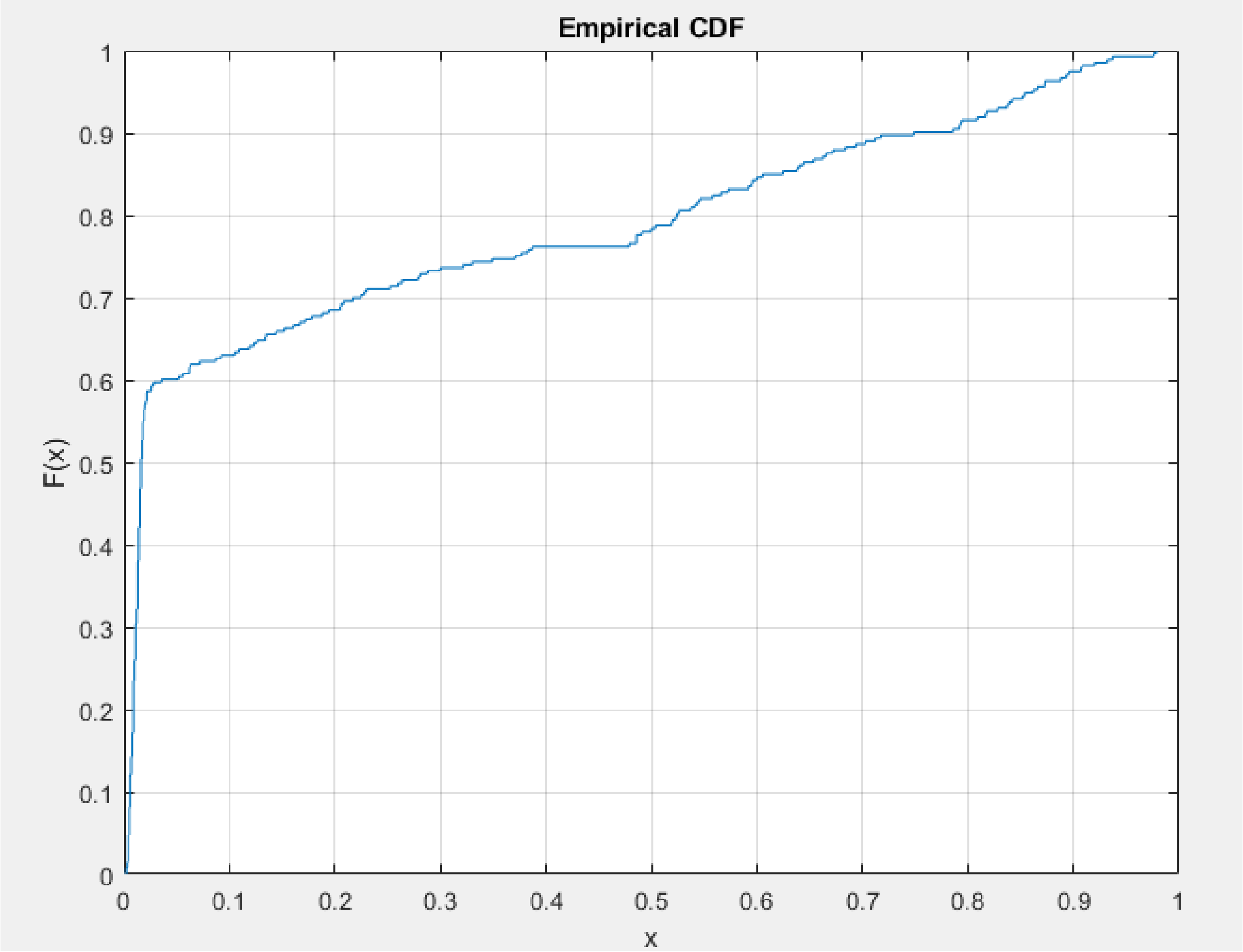
4.3 Feature CDF



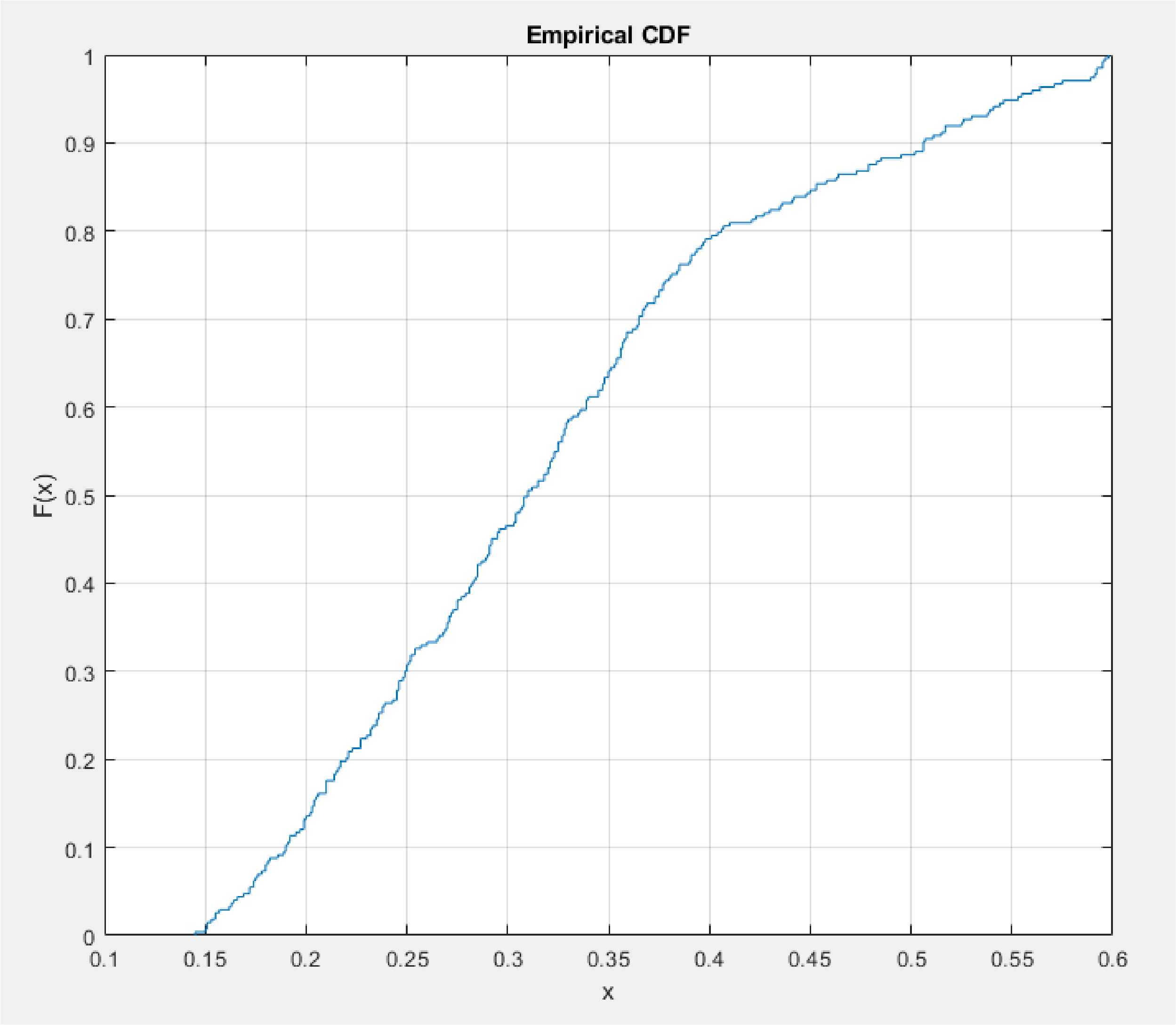
Daily Total Steps



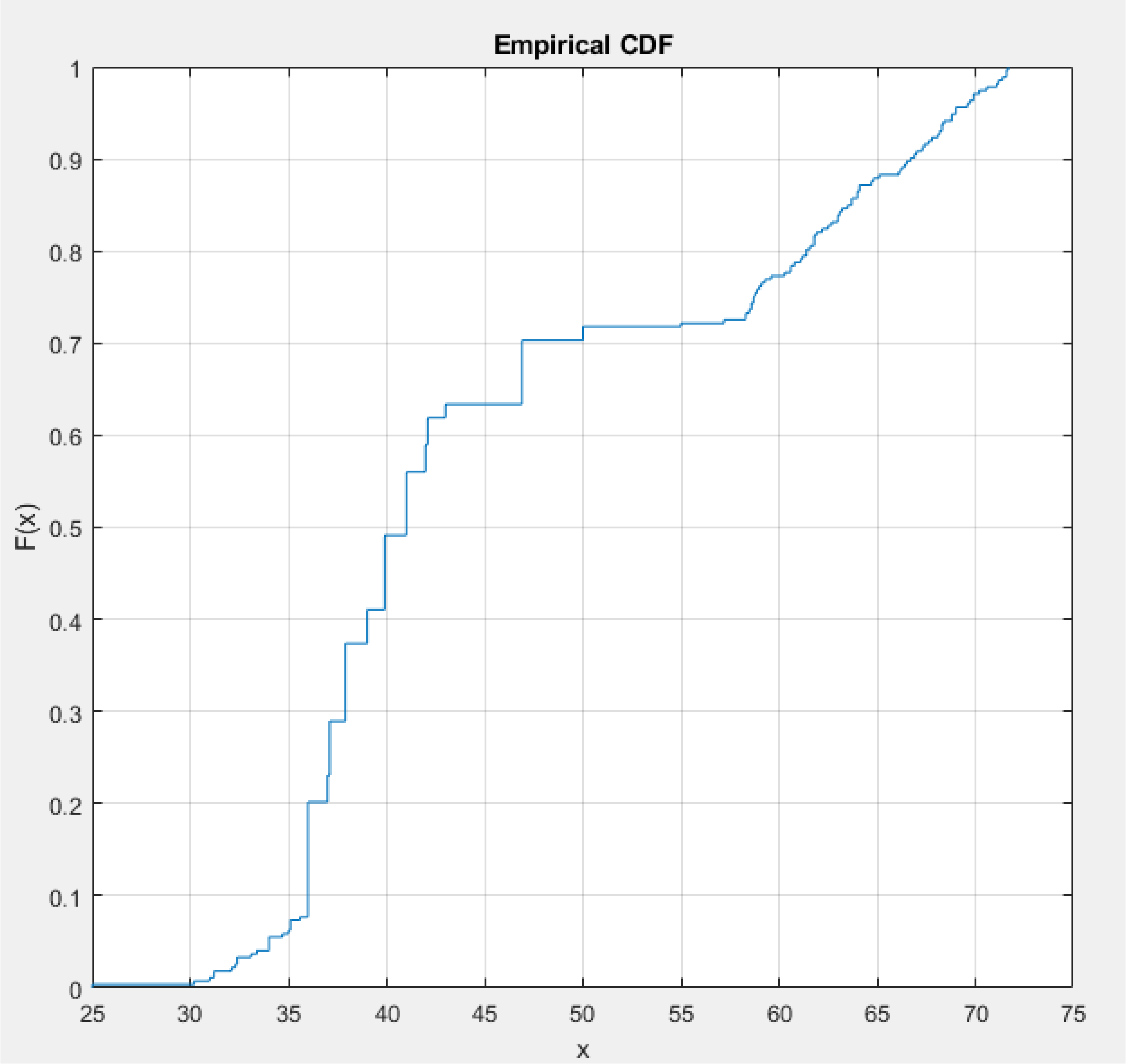
Playing Steps



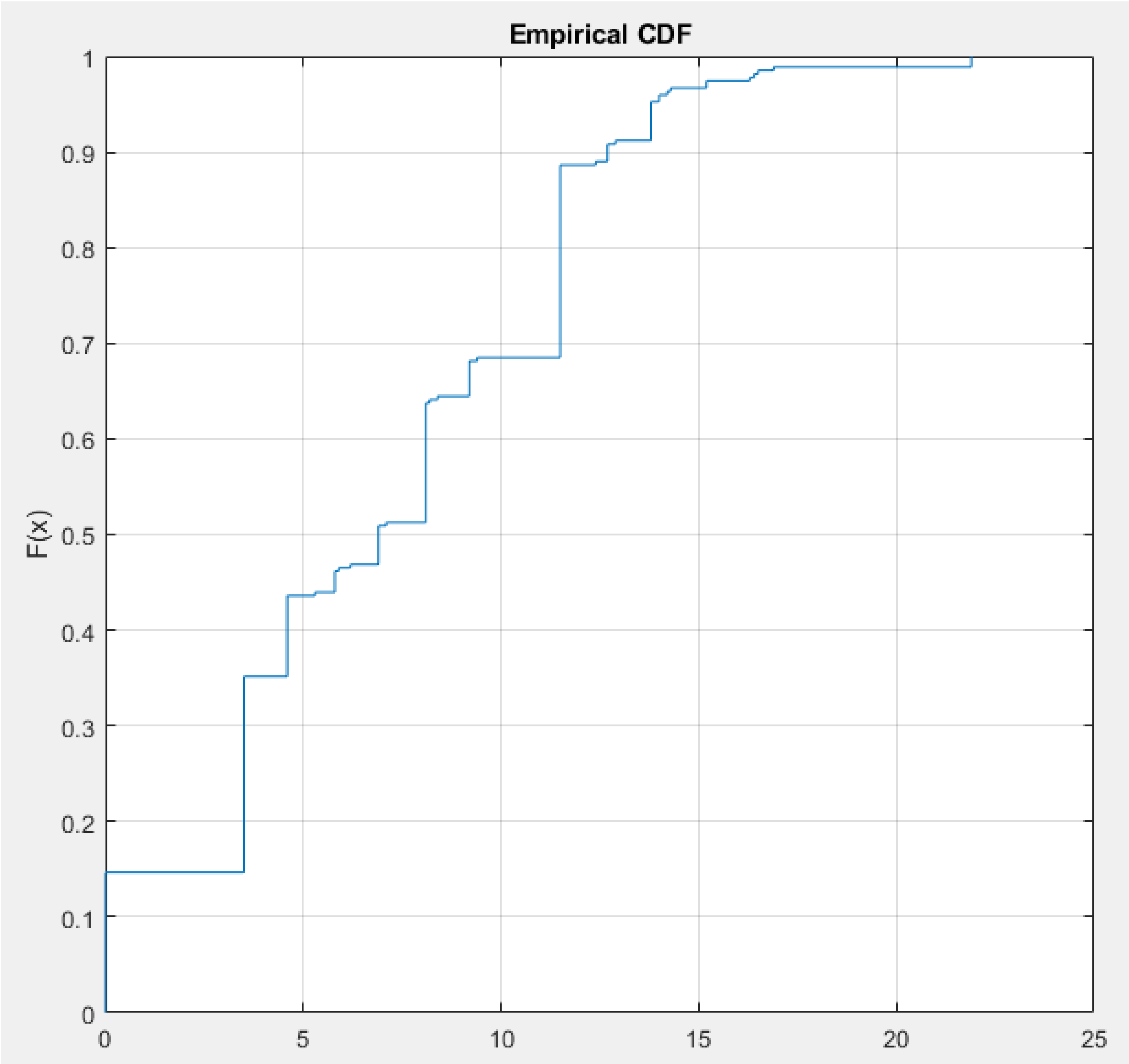
Distance Traveled



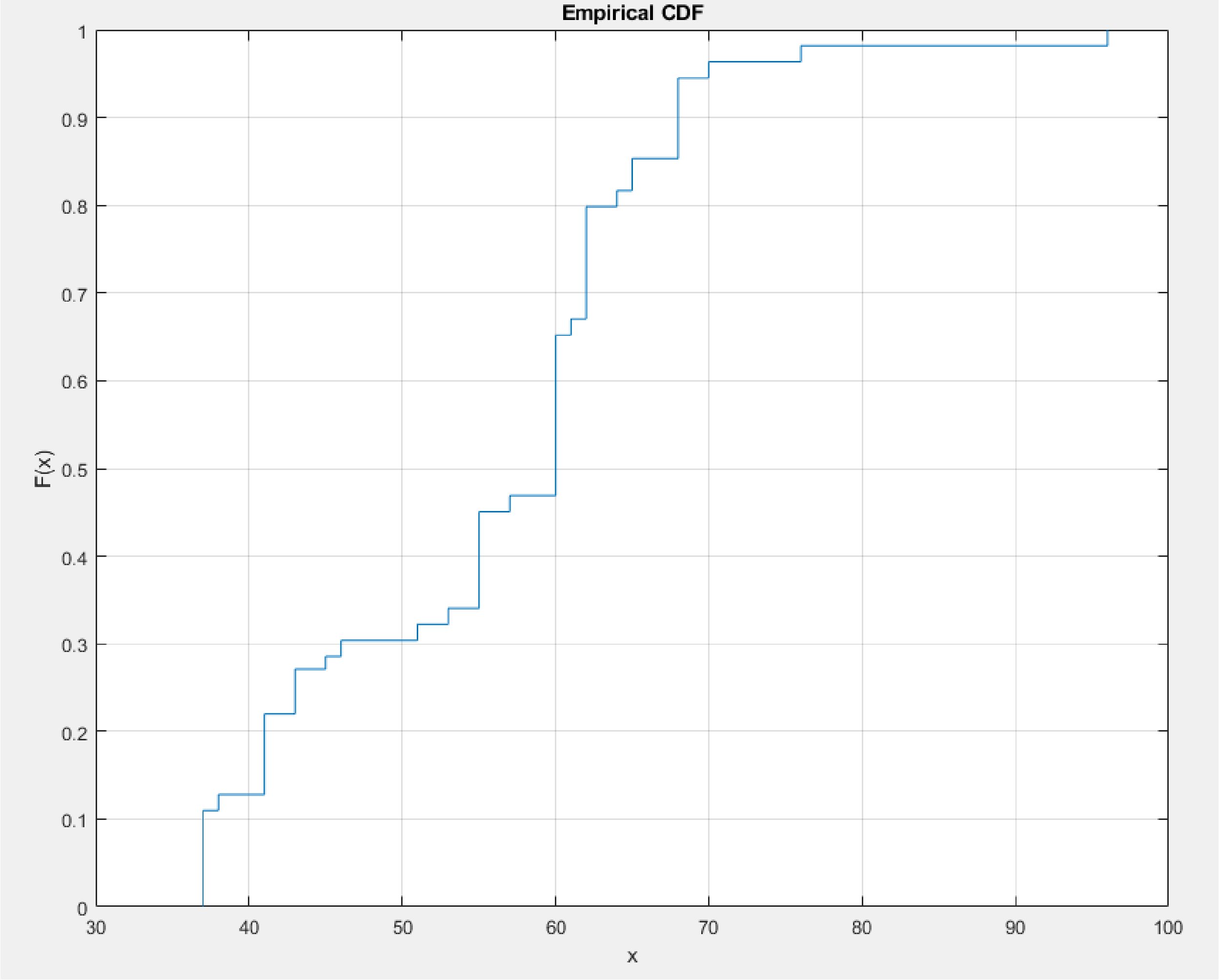
Distance Traveled From Home



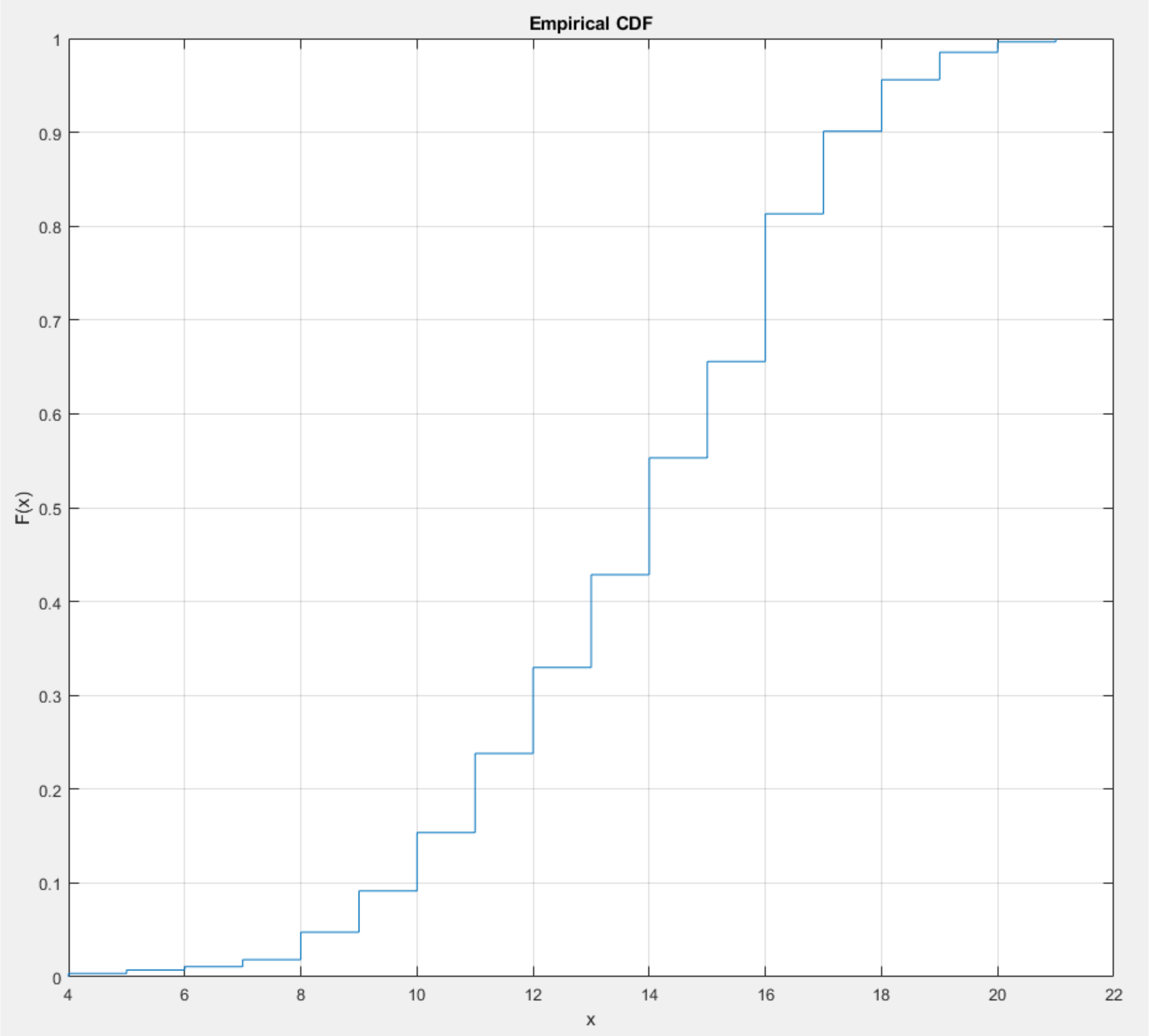
Temperature



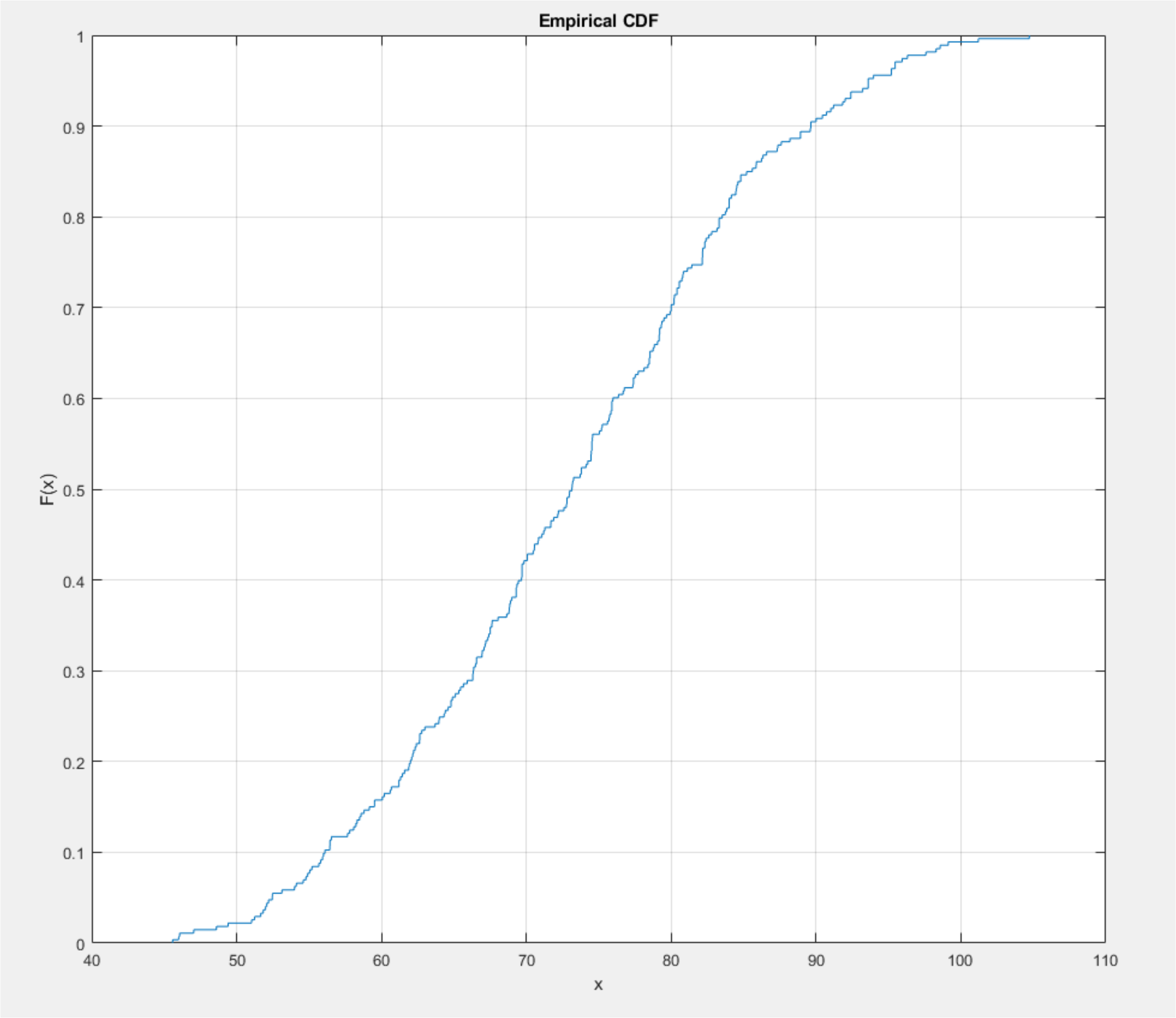
Wind


Empirical CDF
F(x)
x
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
30
40
50
60
70
80
90
100

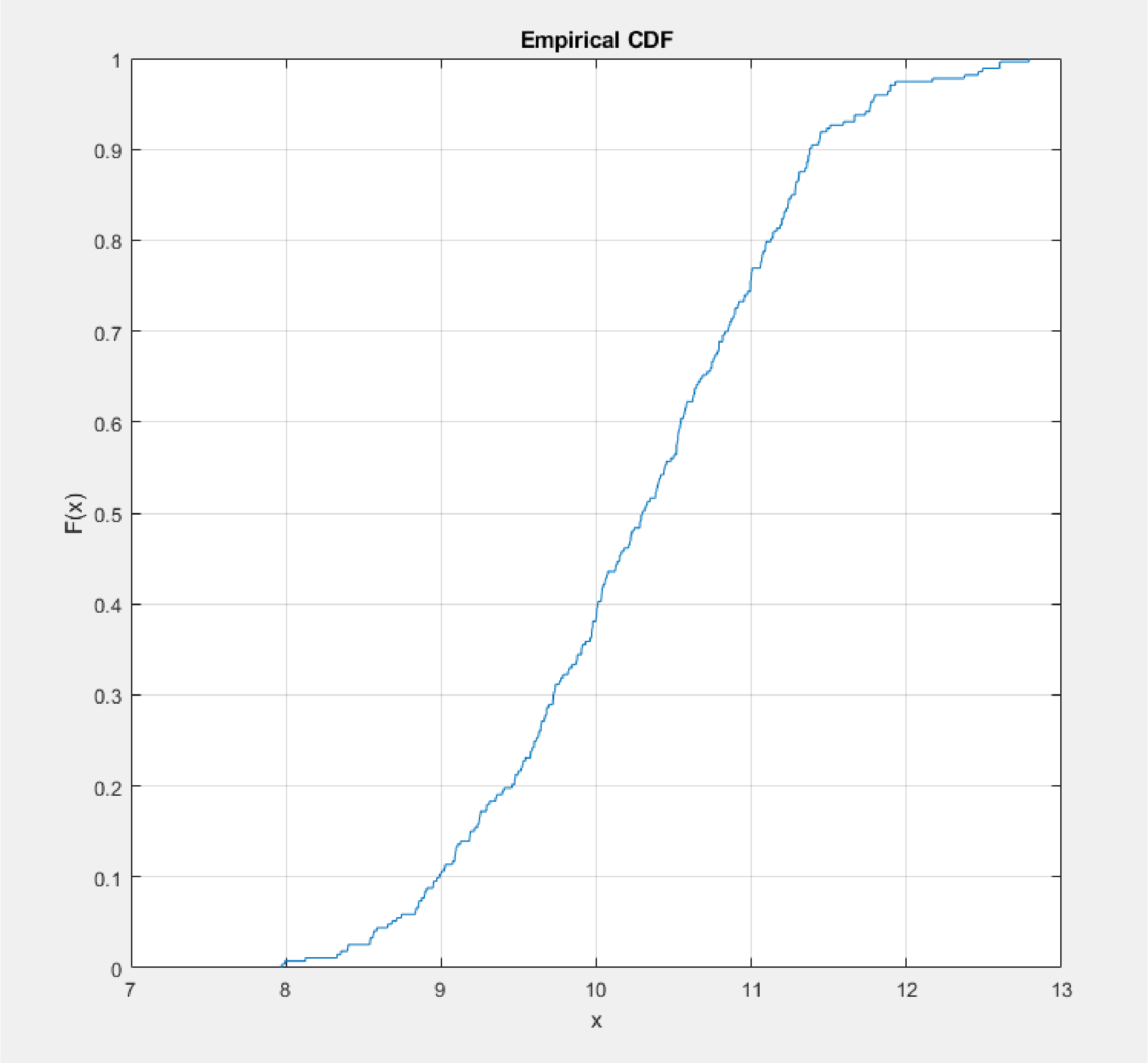
HUmidity



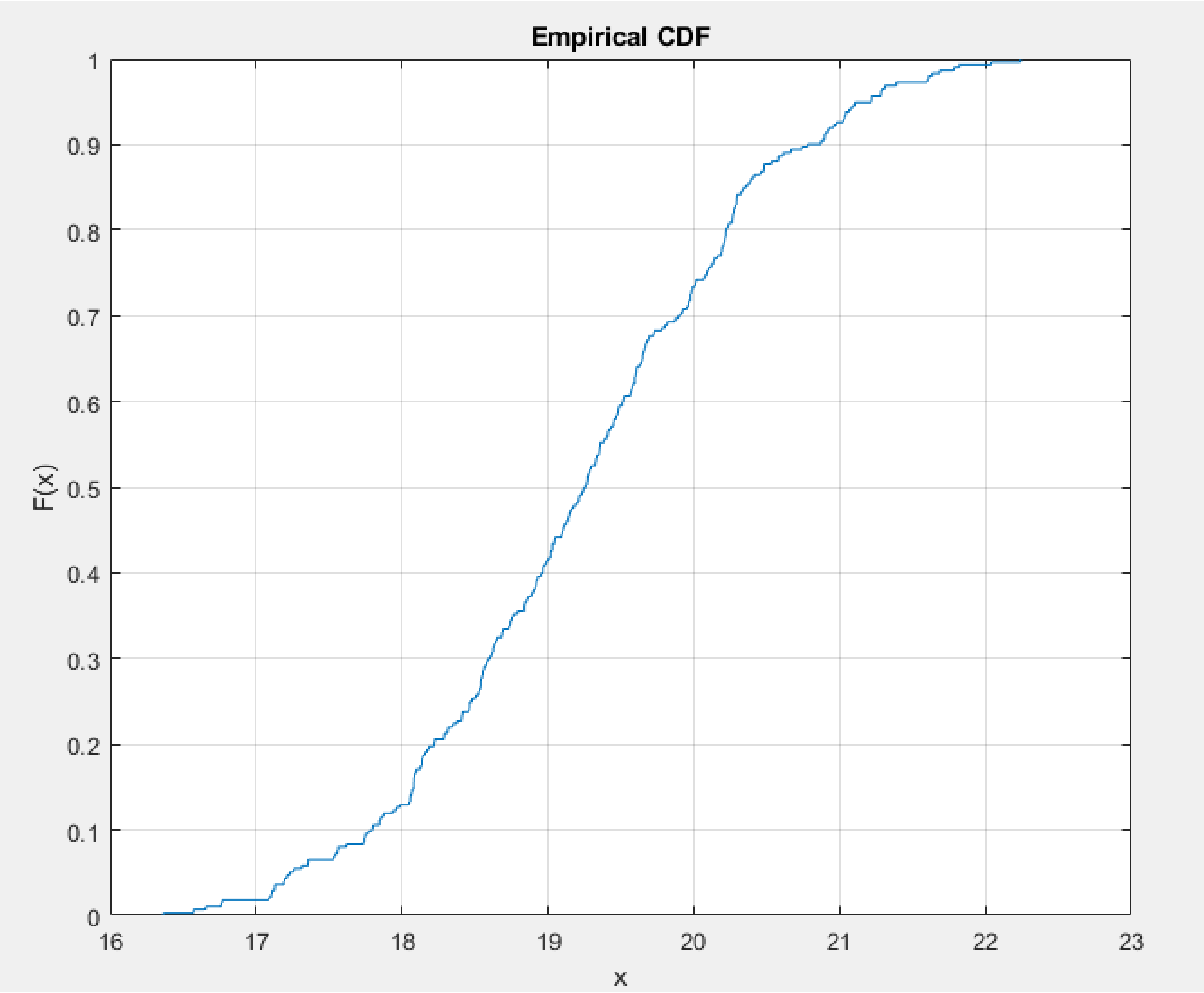
Times App Opened


Empirical CDF
F(x)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
40
50
60
70
80
90
100
110
x

Time Played



Earliest Open



Latest Close