



Using Machine Learning to Identify Potential Problem Gamblers

Jonas Gustafson

Jonas Gustafson

Degree Project in Interaction Technology and Design, 30 ECTS Credits

Spring 2019

Supervisor: Ulf Holmgren

Examiner: Thomas Mejtoft

Master of Science Programme in Interaction Technology and Design, 300 ECTS Credits

Abstract

In modern casinos, personnel exist to advise, or in some cases, order individuals to stop gambling if they are found to be gambling in a destructive way, but what about online gamblers? This thesis evaluated the possibility of using machine learning as a supplement for personnel in real casinos when gambling online. This was done through supervised learning or more specifically, a decision tree algorithm called CART. Studies showed that the majority of problem gamblers would find it helpful to have their behavioral patterns collected to be able to identify their risk of becoming a problem gambler before their problem started. The collected behavioral features were time spent gambling, the rate of won and lost money and the number of deposits made, all these during a specific period of time. An API was implemented for casino platforms to connect to and give collected data about their users, and to receive responses to notify users about their situation. Unfortunately, there were no platforms available to test this on players gambling live. Therefore a web based survey was implemented to test if the API would work as expected. More studies could be conducted in this area, finding more features to convert for computers to understand and implement into the learning algorithm.

Acknowledgements

I want to thank all the co founders at Source Empire, our company, for understanding that I needed to work part time to be able to conduct this thesis, Fredrik Johansson, Martin Willman and Emil Ottosson. Furthermore I would like to thank my supervisor at Umeå University, Ulf Holmgren, for our regular meetings to improve this thesis. Without them I would not be able to conduct the thesis in time. Finally, a big thanks to my peer reviews group, Tonje Lindmark, Cecillia Wikberg and Måns Hellgren for our fun and motivational meetings during the semester.

Contents

1	Introduction	1
1.1	Problem Statement	2
1.2	Objective	2
1.3	Limitations	2
1.4	Source Empire	2
1.5	Brawl Gaming	3
1.6	Abbreviations	3
1.6.1	Gambling	3
1.6.2	Gaming	3
1.6.3	e-sports	3
1.6.4	API	3
1.6.5	Features	3
1.6.6	Labels	3
2	Theory	5
2.1	Machine Learning	5
2.1.1	Types of Learning Algorithms	5
2.1.2	Decision Trees	6
2.1.3	CART	7
2.1.4	Overfitting	8
2.1.5	Pruning	8
2.1.6	Features and Labels	8
2.2	Addiction	10
2.2.1	Pathological Gambling	10
2.2.2	Mood Disorder and Gambling	11
3	Method	13
3.1	Research	13
3.1.1	Literature Study	13
3.1.2	Interviews	13
3.1.3	Survey	14
3.1.4	Choosing Programming Language	16
3.1.5	Choosing Algorithm	16
3.1.6	Choosing Significant Features and Labels	17
3.2	Development	17
3.2.1	Building the Decision Tree	17
3.2.2	Testing the Decision Tree	19
3.2.3	Creating API End Points	19
3.2.4	Developing a Web Based Survey for Self Identification to Test the API	19

3.3	User Testing the Web Survey	20
4	Result	23
4.1	Interviews	23
4.2	Survey	24
4.3	Choosing Significant Features and Labels	26
4.4	Building the Decision Tree	26
4.5	Testing the Decision Tree	27
4.6	Final Result of the Developed Web Survey	31
4.7	User Testing	34
5	Discussion	35
5.1	The Impact a Service Like This Could Have	35
5.2	Interviews	35
5.3	Building and Testing the Decision Tree	35
6	Conclusion	37
6.1	Future Work	37
6.1.1	Storing Data About Users Behaviour	37
6.1.2	Making Decisions for Problem Gamblers	37
6.1.3	Giving Gambling Companies a tag for Responsibility	38
6.1.4	Open Source	38
	References	39
	Appendices	43
A	Main	45
B	Tree	47
C	Decision Node	53
D	Question	55
E	Leaf	57
F	Player/Gambler	59
G	Mail	61

1 Introduction

The term “gambling” has many different connotations to it, but one definition is “to take an action with the hope of a desired result.” This is frequently considered the goal of casino corporations: they aim to lure in general users, not necessarily to create problem gamblers. Problem gambling is now recognized as an addictive disorder where the affected individual has an urge to keep gambling despite negative consequences that may occur. This disorder is very similar to substance abuse disorders; a problem gambler craves the risk to bet money like an alcoholic suffers a need for alcohol [1]. Furthermore, a gambling disorder partially targets the same regions of the brain as alcoholism, including the frontal cortex and similar neurotransmitter systems like the dopaminergic system[2]. Drinking alcohol releases the neurotransmitter dopamine within the human brain, just as gambling does. Although dopamine is released due to chemical intake when consuming alcohol, it is released while gambling when an individual wins something of value and achieves reward over the risk of losing[3][4].

Gambling, in some form, has existed for a long time. In fact, it is believed that playing cards emerged in China in the ninth century[5][6][7]. The first casinos were known as gambling houses, and appeared in Italy in the seventeenth century. The Ridotto, a Venetian gambling house, was established in 1638 to provide a controlled gambling environment. This style of gambling became a trend, and by the nineteenth century, casinos were established throughout Europe[8]. Online gambling first emerged with the creation of the first online casino in 1994. Since then, the market has exploded, with advertisements from various companies emerging everywhere. In modern casinos, personnel exist to advise—or in some cases, order—individuals to stop gambling if they are found to be gambling in destructive methods. Users are at-risk of destructive gambling when participating online, also known as remote gambling [9].

As Grant states[10], there is a close link between gambling addiction and mood disorders [11]. His article states that 25% of all gambling addicts also have bipolar disorder (previously known as manic disorder). Additionally, Grant acknowledges that studies have shown that as many as half of all gambling addicts also suffer from depression. Therefore, this means that most problem gamblers also identify with the category of mood disorders.

One reason why gambling is very addictive is partly due to today’s ease of access. Online casinos are very accessible to individuals familiar with basic technology. The only action necessary to start gambling money online is to deposit money in an online casino. From there, it is possible for the user to bet different amounts of money, including their entire deposit. When the user has run out of deposited money, it is just as easy to start over and deposit more. Therefore, it is critical to be able to detect early problem gamblers in today’s society. According to Monaghan, it is possible to do so by analyzing a user’s gambling behavior [12]. Furthermore, Cloutier states that the number of games played can be significantly reduced if pauses and/or pop-up messages are implemented in the platform

[13]. These solutions would serve to inform users about the risk of gambling too much. This thesis will aim to identify participants with the smallest symptoms stretching to those with a fully developed gambling disorder.

Today's technology offers a variety of algorithms to help computers make decisions. This type of decision-making is referred to as artificial intelligence, and a branch is known as machine learning. With machine learning, there is a probability to help identify problem gamblers and send a notification to them about potential problem gambling. Decision trees could be used to classify a gambler's risk of developing a disorder. This could possibly be executed with the help of the user's own behavior. The behavior could contain data such as money spent on gambling, revenue from gambling, income from work, time spent on gambling per day, and age.

1.1 Problem Statement

Remote gambling can be dangerous because users are often unsupervised. As a result, it is possible for participants to gamble while affected by drugs and alcohol or other influences contributing to a gambling addiction, such as mood-and or bipolar-disorders. Considering the prevalence of gambling problems in modern society, a recent push for gambling corporations to assume larger responsibility and take action has been developing. Therefore, a decision was made to evaluate the possibilities to discover a solution for this problem that could be used by various casinos and similar services.

1.2 Objective

The objective of this thesis is to use machine learning to identify people with a risk of developing a gambling disorder. Specifically, an API will be implemented for gambling platforms to use. The API will determine risk based on a specified scale ranging from zero to five, with a score of five representing the highest substantial risk of a user becoming a problem gambler. The API will then notify at-risk users of its findings via email.

1.3 Limitations

Because no casino platforms were available to connect the API to, there was no way to test the system on players gambling in real time. The problem with the availability was that no casino corporation were willing to give out their source code to try out the API during development. Therefore, to test the API, a web based survey was implemented, which was connected to the API. The survey gathered data by letting users manually fill it in.

1.4 Source Empire

This thesis was conducted at Source Empire, located in Umeå, incubated by Uminova Innovation. Source Empire, founded in early 2018, is a startup that recently developed a safe way for gamers to play skill-based, e-sport games for money. The platform for this methodology is known as Brawl Gaming[14]. The author is one of the four co-founders of the company.

1.5. Brawl Gaming

1.5 Brawl Gaming

As mentioned above, Brawl Gaming is a platform where gamers can compete in skill-based, e-sport games for money[14]. As in poker, money flow occurs after a user places a bet on their own success. When a player wins, the money pot transfers to the winner's account. The major difference between Brawl Gaming and the platform design of a regular casino is that Brawl Gaming's competitions solely depend on skill. The purpose of this service is to enhance the gamer's experience and motivate them to improve their competence.

1.6 Abbreviations

This section will explain a few crucial abbreviations to ease the understanding of the following chapters.

1.6.1 Gambling

Gambling is the activity engaging in risk to either win or lose money based on the result of an activity. It can be performed through a variety of activities, such as playing a computer game against someone or betting money on a horse race.

1.6.2 Gaming

Gaming is the activity of playing video games. It extends from playing a single game on a phone, to a massive multiplayer computer game, to a sports game on a console, such as an X-box or PlayStation.

1.6.3 e-sports

The term "e-sports" is simply a shortened word for electronic sports, and refers to playing skill-based competitive computer games. The competitive video game phenomenon has grown to become a worldwide, multi-billion-dollar industry. There are millions of fans following tournaments on a regular basis.

1.6.4 API

API stands for Application Programming Interface. An API is the part of a server where requests are sent to by clients, and responses are sent back.

1.6.5 Features

In machine learning, features are the inputs that are given to the system to make a decision. For example, the features of trees could be bark color, leaf form and height. Depending on what the features are, the system will make a decision on what type of tree it is.

1.6.6 Labels

Labels are the desired output when training a machine learning system. From the example presented in 1.6.5, a tree labeled with the tree type birch could have features like white bark color, small pointy leaf form and a tree height of 20 meters. This lets the system know how to identify a birch.

2 Theory

This chapter will explain different nouns to make the report easier to understand. It will cover the technical parts, as well as psychological conditions covering ordinary behaviors amongst problem gamblers. Finally, it will cover the differences between lottery and competition.

2.1 Machine Learning

As mentioned previously, machine learning is a branch of artificial intelligence[15]. It is based on the understanding that systems will learn by processing given information. The purpose of machine learning is to identify patterns and determine outcomes with little to no human interference. Machine learning algorithms create a model of sample data, or training data. This data is used to make decisions without the system being explicitly programmed to perform a specific task [16]. In fact, multiple algorithms exist within machine learning that are used for various purposes. The algorithms differ in their approach, output, and solving intent.

2.1.1 Types of Learning Algorithms

When discussing machine learning, it is important to understand three main learning categories, known as supervised, unsupervised, and reinforcement learning. This section will introduce all of them and give examples on situations where each of them are fit to be used and why. For the purposes of this thesis, supervised learning will be the primary learning system of discussion. Ultimately, this section will give an understanding of why supervised learning was chosen for the purpose of the thesis.

Supervised Learning

Supervised learning occurs when algorithms learn from interpreting labeled data. Both input and output data are provided to the algorithm. The data is labeled for classification with the purpose of providing a learning basis for future data processing. The algorithm is provided with training data to assist in future decision-making. The training data includes inputs with desired outputs. An example of supervised learning is an image-processing algorithm designed to recognize images of cats. The input photos comprised of other animals would result in an output telling the algorithm that no cats are recognized in the photo, and vice versa. This type of learning is considered supervised because humans ultimately retain control of what conclusions should be drawn from the algorithms. Supervised learning can be further broken down into two categories, known as classification and regression. Classification predicts what category provided data should belong to, such as in animal species detection, where cats and dogs belong to two different categories. It is also capable of finding interactions between features. For instance, looking at figure 2.1, the system would be able to predict where a person would go to eat depending on the features hunger and

amount of money. On the other hand, regression is used to forecast numerical values based on previously observed data, which can be applied to predict stock market changes [17]. One advantage that supervised learning has over unsupervised learning is that the system is likely to draw conclusions that humans can relate to, because humans have provided the algorithm with the basis for decision-making[18].

Unsupervised Learning

Unsupervised learning is implemented so that the system learns from taking real-time action. Consider a robot whose task is to place an item in a specific location. If the robot places the item in an incorrect area, it will be notified that it made an error, and this will cause the robot to refrain from placing an item in that specific area again. Hence, the robot learns from its prior experience. Chess is good example where unsupervised learning would be more efficient than supervised learning. Using supervised learning, a human would have to present the learning system with every move to take depending on what the opponents action is. Instead, using unsupervised learning, the system will learn what not to do, each time it loses a game. Practically, the system will make random actions at the start of the learning process and learn to make better decisions from each move. Unsupervised learning is therefore dependant on knowing when something good or bad has happened. In chess, the learning system would therefore benefit from knowing the value of each chess piece. Moving into the subject of this thesis, there is no data available on what is good or bad. Furthermore, being able to identify interactions between features are required, which is not possible using unsupervised learning. Therefore, unsupervised learning could not be chosen to conduct this thesis.

Reinforcement Learning

Reinforcement learning is a type of Machine learning where the agent learns from its environment by taking actions and observing the results. The method of learning from interacting with the environment is taken from natural human experiences. Take for example a baby seeing a fireplace. The interest of the fireplace makes the baby approach it. The warmth of the fire gives a positive feeling. The baby understands that the fire is a positive thing to sit nearby. The baby touches the fire and it burns. Hence, a negative feeling. An understanding of how to interact with fire emerged through the baby's actions. This is how reinforcement learning works. Points for positive feelings is gathered through for example +1 and the representation for negative feelings are for example -1. The goal is to reach as many positive points as possible [19]. Since there are no environments to interact with, reinforcement learning was ruled out for the purpose of this thesis.

2.1.2 Decision Trees

A decision tree is a supervised learning method that is considered non-parametric. This means that the data is not required to fit a normal distribution. Therefore, decision trees can handle heterogeneous data, or different types of data, better than many other methods of supervised learning. A decision tree could resemble a flowchart where each node represents a test. They are typically illustrated upside-down, with the root node at the top of the chart, followed by a number of branches and decision nodes leading to leaf nodes at the bottom. Each test has a number of outcomes associated with certain chances of occurrences. These outcomes are the branches. A branch can lead to a new node that will result in a new test,

2.1. Machine Learning

or the branch leads to a leaf. Every leaf in the decision tree represents a class label, which equates with a decision. The full path from root to leaf represents a number of decisions to determine the final outcome. The process of determining which node path to follow is typically determined by answering yes or no questions, but could also lead to multiple outcomes. See Figure 2.1 for a graphical explanation.

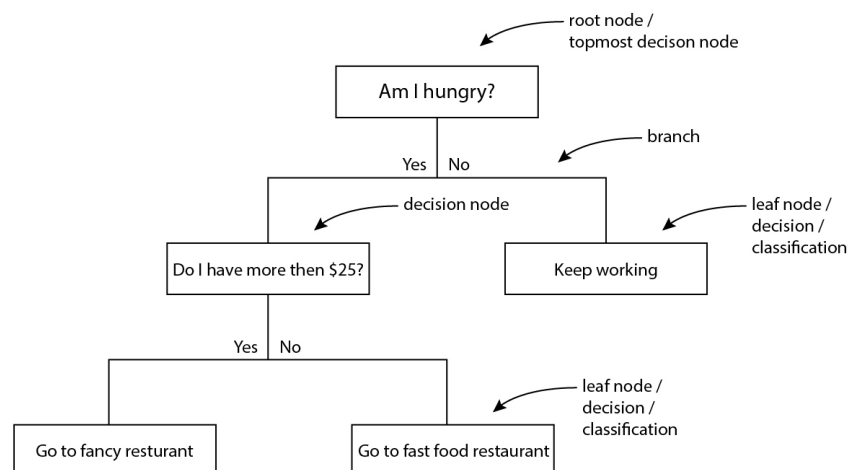


Figure 2.1: A simple decision tree determining whether to eat lunch or not, and depending on the amount of money owned, it also determines where to eat.

2.1.3 CART

CART is a form of decision trees, using supervised learning. In 1984, Leo Breiman, Jerome Friedman, Richard Olshen, and Charles Stone developed a method of predictive machine learning known as classification and regression trees, or CART[20]. These decision trees are used to construct prediction models from data. By recursively partitioning the data and associating a prediction model within each partition, the model can be represented graphically as a decision tree[21]. In other words, CART algorithms repeatedly identify the predictor variable with the most appropriate match to divide the data into two subsets. The algorithm will keep doing this until there is only one or few data points per leaf that remain in the decision tree. The partitioning is conducted by using the Gini-coefficient, also known as the Gini index. The Gini index acquired its name from Corrado Gini and is used to measure inequality in a distribution. More specifically, it is used to measure how far a distribution deviates from a perfectly equal distribution. In machine learning, the Gini index is used to ask the most relevant question for partitioning information into two new branches.

The CART algorithm begins with the root node, which contains a set of possible given elements. After a split, if all elements are of the same class, a leaf node is created and no more deviations will be possible on that set. If not all elements are of the same class, the algorithm will identify the most relevant feature to split for each node. For example, Figure 2.2 illustrates how weather is likely to affect an individual's decision to play golf. As shown in the figure, if the temperature is cold, a person will most likely not play golf, regardless of the wind or humidity. Therefore, a relevant first split would partition temperature into two new nodes where all warm days are organized in one path and all cold or mild days are arranged in the other. To see the fully trained tree using the CART algorithm, see Figure 2.3

2.1.4 Overfitting

The phenomenon of overfitting occurs when a model does not generalize well enough from the training data to apply to unseen data. In other words, the model has learned the “noise” instead of the “signal.” The signal refers to the expected processing of training data that leads to the correct decision. However, noise refers to irrelevant information not needed to make a decision. Therefore, noise should not be learned. A model that learns from the noise is considered overfit because, although it fits the training data well, unseen data is poorly managed. Methods exist to overcome overfitting, and for decision trees, a method called pruning is used [22].

2.1.5 Pruning

Pruning mechanisms are used to prevent overfitting. In pruning, the training data is randomly divided into two groups. One group contains the new training data, while the other holds the validation data. A decision tree is fully developed with the help of the training data. Then, the validation data is used to prune the tree. As a result, this is referred to as cross-validation, which is a determinant of the accuracy of the decision tree. Another strategy in pruning is called minimum error, and is used to prune back the tree to the point where the cross-validated error is minimal[23].

2.1.6 Features and Labels

Features, or feature variables, are the attributes that describe an instance[24]. Examples of features include variables that influence an individual to decide whether or not to play golf, such as consideration of temperature, humidity, and wind. Additionally, features that determine types of fruit include color, shape, and weight. When training a machine learning system, features are paired together with labels, which are viewed as the answers, or results. For example, if a person decides it is a good day to play golf, or if they have determined a fruit is an orange or a banana, they have arrived at a label. Another name for a label is a class, see Figure 2.2 for an illustrative explanation. After the system has been trained on the data from Figure 2.2, when informed of the current weather conditions, the system should be able to process and determine if it is indeed a good day to play golf.

2.1. Machine Learning

Features			Label
Temperature	Wind	Humidity	
hot	false	high	Play
cold	true	low	Do not play
hot	false	normal	Play
mild	false	low	Play
mild	true	low	Do not play
mild	true	high	Do not play
hot	true	high	Play
cold	false	high	Do not play
cold	false	low	Play
hot	true	normal	Play
cold	false	normal	Play
mild	true	high	Do not play
hot	true	normal	Play
cold	false	normal	Play
cold	false	high	Do not play
mild	true	high	Do not Play
hot	false	low	Play
hot	false	normal	Play
cold	true	low	Do not Play

Figure 2.2: Features and labels that could be used to train a system to decide if to play golf or not depending on weather conditions.

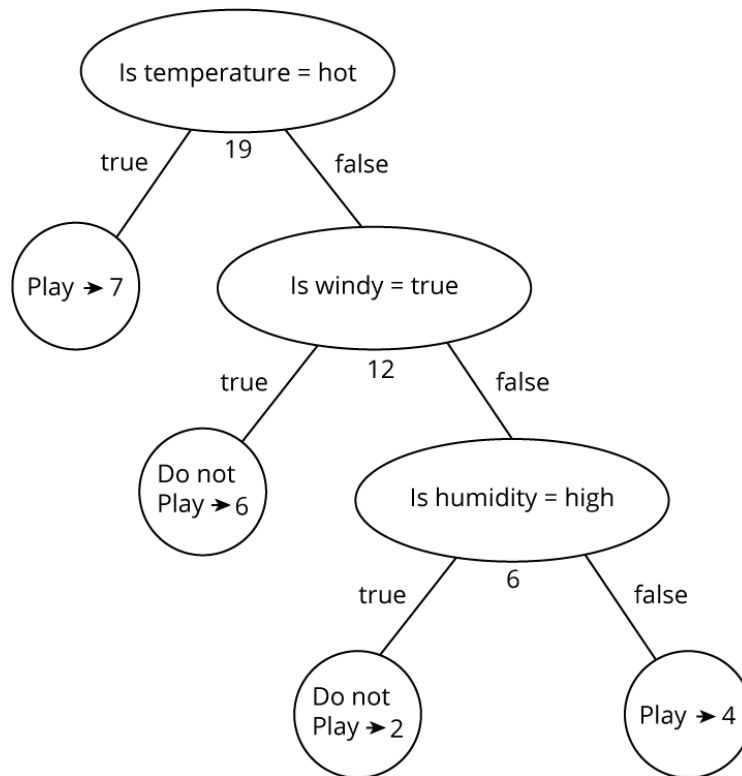


Figure 2.3: Decision tree made with CART using the information from figure 2.3. The decision tree represents whether a person will play golf or not, depending on weather conditions.

2.2 Addiction

An addiction occurs when a person engages in a behavior where the outcome results in a compelling motivation to keep pursuing the behavior, even if it leads to destructive consequences. There are many different behaviors or substances in which a person could become addicted to. Among these are alcohol, cocaine, nicotine, heroine, and behaviors like gambling, gaming, and shopping. Addictive substances and behaviors share a fundamental addictive element: both intensely trigger reward and reinforcement brain pathways[25]. Many of these brain pathways involve dopamine, which produces motivational salience [26]. Motivational salience represents a cognitive process and a form of consciousness that influences an individual's behavior towards or away from a task or an outcome. It adjusts the magnitude of behaviors that eases the attainment of a specific goal and the amount of risk a person is willing to take to achieve it[27]. Since addiction affects the executive functions of the brain, people who are affected by addiction may not be aware that their behavior causes problems to themselves or others.

2.2.1 Pathological Gambling

A pathological gambler is an individual that succumbs to the impulse to bet money, which often leads to substantial personal and social consequences. The need to gamble develops into a stressful behavior that can only be satisfied by further gambling. Over the course of a twelve-month period, the appearance of nine symptoms is often used to identify an indi-

2.2. *Addiction*

vidual with a gambling disorder. The presence of four to five symptoms is often associated with a mild gambling problem, while six to seven symptoms represents a moderate problem, with a severe gambling problem comprising of eight to nine symptoms. The symptoms are as follows[28]:

- Needing to gamble progressively larger amounts of money to feel the same or more excitement.
- Having made many unsuccessful attempts to cut back or quit gambling.
- Feeling restless or irritable when trying to cut back or quit gambling.
- Preoccupation or excessive thoughts (e.g., previous gambling experiences, planning the next gambling venture, ways to get money to gamble with again).
- Gambling increases, and may even occur to escape problems/feeling distressed (feeling helpless or guilty), or feelings of sadness, or anxiety are present.
- Gambling larger amounts of money to try to recoup previous losses (chasing previous gambling losses).
- Lying about the amount of time or money spent gambling.
- Losing a job, relationship, or educational or career opportunity due to gambling.
- Relies on others to borrow money to get by due to gambling losses, especially when financial situations become desperate due to involvement in gambling.

2.2.2 **Mood Disorder and Gambling**

Mood disorders includes depression and bipolar disorders. A bipolar disorder can cause mood swings, fluctuating energy levels, and affects the ability to get things done. It is a manic-depressive illness, and researchers believe that during depressive episodes, some individuals turn to gambling to self-medicate [29][30]. This means that they use gambling to speed up the depressive episode. During manic episodes, people often have impulse-control issues. Therefore, it is believed that gambling might serve as another channel for impulsive behavior[29].

3 Method

This chapter will describe the working process. It will explain why different technical aspects were chosen for this project and how they were used. First, the research will be discussed and sub categorized into literature studies and interviews. Next, the development process will be addressed. This section includes choosing algorithms, programming language, and creating features and labels for the machine learning.

3.1 Research

This section provides details about how information was obtained throughout the project. As mentioned previously, three parts comprise this portion, which include literature study, interviews, and development. The literature study includes how information was retrieved from different types of literature. Information derived from expert interviews will be presented, and the developmental aspect will introduce what techniques that were used to create the machine learning system. Furthermore, it will incorporate how labels and significant features

3.1.1 Literature Study

In order to develop an understanding for machine learning and how it is used in different scenarios, an expansive literature study was conducted. The study intended to locate information about similar cases where machine learning algorithms were used to identify problem gamblers. Due to the fact that the subject of online gambling and how to prevent it, is relatively new, few studies have been previously conducted. As a result, reliable websites with writers educated in subjects of psychology and AI were used as source material, along with relevant articles and reports. To locate this material, Google Scholar was used and references from various other articles were selected [31]. Search words used included “machine learning,” “decision trees,” “pathological gambling,” “identifying problem gamblers,” and “helping problem gamblers,” amongst others.

3.1.2 Interviews

Interviews were conducted to learn more about typical patterns in behavior with problem gamblers. The conducted interviews were solely held with experts for the purpose of getting qualitative information about problem gambling, and to be able to choose the most significant behavioral patterns for the decision tree. A number of emails were sent to organizations concerning gambling disorders to get in touch with experts. Because the author is the co-founder of the corporation ordering this thesis, it was important to explicitly communicate which company it was being conducted for. The reason for this was that people working with preventative pathological gambling were presumed to be hesitant to take part in the interviews for Source Empire. However, interview questions were designed to prompt the interviewee to provide thorough answers that explain in detail how they care for problem

gamblers, and how they decide the extensiveness of an addiction problem. The questions asked were as follows:

- What are typical behavioural patterns amongst problem gamblers?
- How does corporations that exist to prevent pathological gambling, help problem gamblers to break their behavioural patterns?
- How does the identification process of how comprehensive someone's gambling addiction is work?
- Is there an existing scale that explain how extensive someones problem is?

3.1.3 Survey

When the interviews had been conducted, a survey was created using Google Forms. The purpose was to ask users about their opinion on what the reason that a game feels addictive is, specifically when money is involved. The survey was published in several groups on Facebook and reddit where problem gambling was the main topic. People that were openly having a problem as well as those that now days have gotten it under control was contacted to answer the survey. People filling in the survey had previous experience in this specific field. The goal of the survey was to make sure that people with gambling problems would be okay with gambling organizations tracking their behaviours to be able to notify or ban people depending on their actions, see figure 3.1. Because people filling out the form had prior knowledge about problem gambling, the survey would most likely give a hint of what feature to put the most focus into when creating the decision tree later on. The second part of the form would only target people who actually have had a personal problem with gambling, see figure 3.2. The reason for this was to get a notion of how they would feel about getting notifications about their potential problem, how they would feel about getting banned for playing to much and lastly, how they would feel about getting recommendations on how to get help. The reason for conducting this survey was to make sure that problem gamblers actually would find it helpful with a system giving them advice on how to break their bad habits existed.

Survey

Using Machine Learning to Identify Potential Problem Gamblers

Intro

My name is Jonas Gustafson and I am conducting a thesis about pathological gambling. More specifically, about how to use machine learning to identify potential problem gamblers before they develop a severe condition of gambling disorder.

This survey is created to identify the most significant behavioral patterns inhabited by potential problem gamblers.

When filling in this form, you will be anonymous. In no way will anyone ever be able to know what your answers were. The information gathered will be used to get a better understanding of how the early stages of a problem gambling behavior are developed but also, how the behavioral patterns of someone with a severe condition. The information you give will not be stored until you submit the form and it is possible to exit the survey at any time before submitting.

Behavioral Patterns

This section will ask questions about problem gamblers behavioral patterns when gambling.

The behaviors stated in this section are behaviors that could be noticed by someone observing a gambler remotely/online. Therefore, behaviors, like not spending time with your children or not having time to cook, will not be taken in regard.

What type of behavior would you say is the most critical for a problem gambler?

- ☐ *Gambling most of the time awake.*
- ☐ *Spending lots of money no matter how much money that has been lost.*
- ☐ *Depositing money every time the account balance is close to zero.*
- ☐ *A large number of gambling-sessions each day, most of the time awake.*

If you have another behavior in mind, that is simply based on information, please, do share. If you think that this behavior is more crucial then the stated above, please share that as well.

Your answer goes here...

Figure 3.1

Survey

Using Machine Learning to Identify Potential Problem Gamblers

Your gambling problem experience

Would you say that you've had a gambling problem at some point?

- ☐ Yes
☐ No

If Yes

Would you be comfortable to get notification about your level of risk to develop a problem?

- ☐ Yes
☐ No

Would you reduce your gambling if you felt like gambling organizations actually cared about your wellbeing rather than your money?

- ☐ Yes
☐ No

Would it be okay for you to be temporarily banned from all gambling sites if you were identified as having a substantial gambling problem?

- ☐ Yes
☐ No

Would you seek help through a notification given to you with information on who to contact and why the gambling organization thinks you should seek help?

- ☐ Yes
☐ No

Figure 3.2

3.1.4 Choosing Programming Language

With thorough documentation comes great readability, meaning that a language with extensive documentation about machine learning would assist in the development process. Choosing programming language was of utmost importance because documentation was needed to develop the machine learning system. R and Python are considered the most popular languages to develop machine learning systems. Python is known to be flexible and user-friendly, and R is renowned for handling and breaking down statistical data well[32]. Due to its ease of use and its extensive documentation regarding machine learning, Python was chosen.

3.1.5 Choosing Algorithm

In machine learning, there are a multitude of different algorithms to choose from. The process of choosing an algorithm was conducted through researching the options, as well as

3.2. Development

the advantages and disadvantages of each. When selecting the most suitable algorithm, an article specifically written to aid in decision-making was used[33]. When searching for the most suitable algorithm for the system in study, several results were identified. Amongst these were Q-learning, linear regression, and K-means. The final decision was made using a decision tree algorithm because of the attainable amount of understanding required, and because of the simplicity of the data gathered to train the system. Decision trees also contain a number of algorithms. The algorithms considered when making the choice included ID3[34], C4.5[35], CART, and Random Forest[36]. ID3 and C4.5 were quickly eliminated from consideration because of associated disadvantages when dealing with small quantities of data. The choice then stood between Random Forest and CART. Random Forest is considered to be one of the most accurate learning algorithms available; nevertheless, CART was chosen because classifications made by Random Forest can be difficult to interpret by humans. Another benefit of CART is that it establishes interactions among variables, which makes it very suitable for identifying features of problem gamblers.

3.1.6 Choosing Significant Features and Labels

The choice of features is very important when dealing with machine learning. Choosing irrelevant features will result in consequences when the system makes decisions. Therefore, after developing the ML-system, different features were tested to decide what resulted in the most accurate classifications. The labels represented the severity of the gambling problem the user experienced. Therefore, a scale from zero to five was used with each step having an explanatory text to define the number and communicate what it meant to the user.

3.2 Development

This section will include information about how to develop the system that was created during this thesis.

3.2.1 Building the Decision Tree

To start developing the machine learning system, a video series from Google developers was followed to help understand the process[37]. A large amount of code was provided in order to begin working[38]. Using the information collected from the literature study[39] and through interviews, data was created for the system to be able to build the decision tree. By generating fictive gamblers with random data within a certain span for each label, a list of approximately 20,000 players was made. A random player could be generated as follows:

```
Player: {  
    hours-spent: random(0.0 - 4.0),  
    number-deposits: random(0 - 2),  
    win-loss-rate: random(0.8 - 1.2),  
    label: 0,  
}
```

This means that a player who spent zero to four hours a day gambling, deposits money zero to two times, and has a win/loss rate of 0.8-1.2, will not trigger a result identifying a gambling problem because features with this data are labeled with zero. When designing the decision tree, it was important to find multiple combinations of features to result in a

variety of different outputs. For example, eight hours of gambling and losing 5% of the current balance might yield the same result as one hour of gambling and 40% of the current balance lost. This means that high quantities of data were required for the project. To make the training data look as authentic as possible, some random inputs were incorporated. A function was created to make this possible. The function parameters are listed in section 4.3. Because problem gambling can arise in different forms, it is important to provide the system with as many different scenarios as possible. An illustration of how the fictive users were generated can be seen in figure 3.3 and the function used to generate them is shown in figure 3.4.

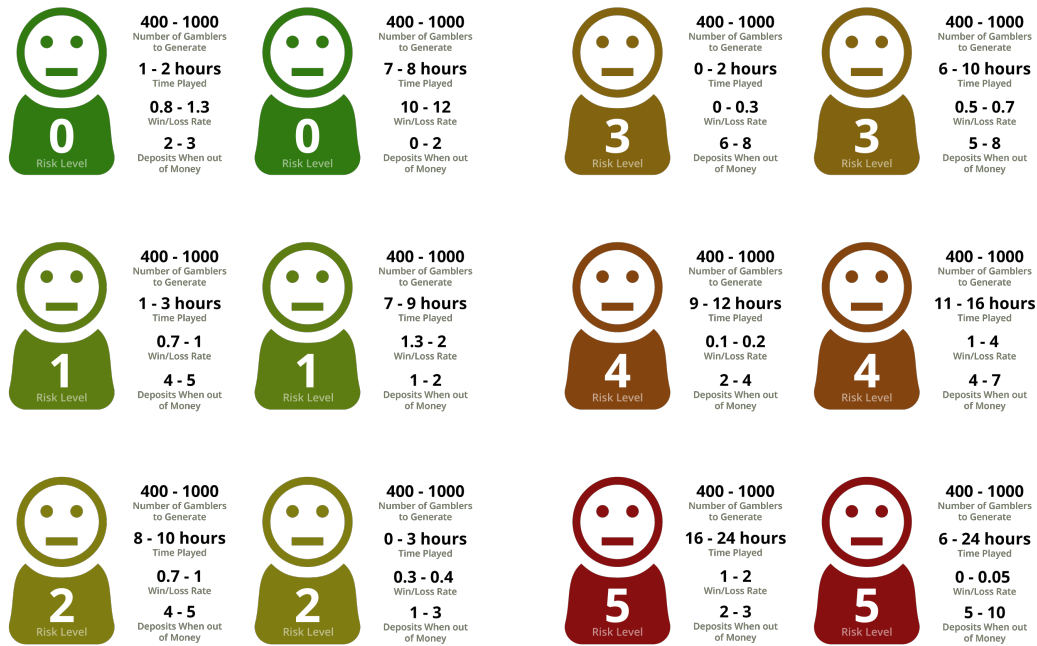


Figure 3.3: This is just an example of how fictive users were generated. Nevertheless, the figure is shown to give an understanding of how the parameters interact with each other. For example, a user playing nine hours could be given a risk level of zero through five. This is where multiple parameters comes in handy. Someone that actually earns a big amount of money might be pursuing gambling as a full-time job, therefore nine hours might not be a too great amount of time spent. Although, someone losing the most part of all money that has been deposited, might have a big problem, spending nine hours a day, never to win anything.

```
def create_player_list(iteration_span, time_span, win_rate_span, nr_deposit, answer):
    player_list = []
    for x in range(random.randint(iteration_span[0], iteration_span[1])):
        player_list.append(
            Player(random.randint(time_span[0], time_span[1]),
                  random.uniform(win_rate_span[0], win_rate_span[1]),
                  random.randint(nr_deposit[0], nr_deposit[1]),
                  answer).get_player_params())
    return player_list
```

Figure 3.4: Function to generate a list of players with features amongst the different spans provided.

3.2. Development

3.2.2 Testing the Decision Tree

To test the decision tree, the only action needed was to inset chosen features while knowing what output that should emerge. If the decision tree produces the wrong output, more accurate features or a larger number of elements included when building the tree are needed. In Figure 3.5, an actual test was made to determine the accuracy of the decision tree (one for each level in the scale from zero to five). It shows that five out of six were correct. This means that the tree needs to be more accurate when identifying the highest possible level of problem gambling, and so the tree was built again. Normally when evaluating a decision tree, a significantly larger amount of tests should be made, which will be shown in the results. A goal of 95% accuracy was set, as well as the avoidance of any large differences in the testing (ex. a zero getting classified as a five).

Test #	Time Spent Gambling	Win / Loss Rate	Number of Deposits When out of Money	Actual Result	Predicted Result	Is Correct
1	1 hour	1	1	0	0	Yes
2	4 hours	1	4	1	1	Yes
3	4 hours	0.5	2	2	2	Yes
4	9 hours	0.55	7	3	3	Yes
5	7 hours	0.32	7	4	4	Yes
6	17 hours	1.3	2	5	4	No

Figure 3.5: These are the results of a simple test conducted in the early stages of the project, it shows that it could guess the correct level on five out of six tests.

3.2.3 Creating API End Points

Because the information given to users is very limited, the number of end points would not need to be great. An end point is the place where APIs send requests and where it stores its resources[40]. Users need to acquire the information about their gambling and, therefore, the only endpoint created for the purpose of giving that information was a request on the URL ending with “/problem-gambling,” yielding a response explaining a user’s gambling situation.

3.2.4 Developing a Web Based Survey for Self Identification to Test the API

At the time of development, no service or platform was available to connect the API with. Therefore, a website was developed where users were able to answer questions about their problem gaming themselves.

The reason for creating the web survey was solely to make sure that the system could receive requests and produce a relevant answer to the users conducting the test on the website. When choosing colors and design patterns for the website, brief research was conducted regarding color theory and human responses to color. The shade of blue is said to symbolize trust, truth, and stability[41]. These factors are important when discussing health care matters with users. The website was intended to elicit feelings of trustworthiness within the user, as well as to give the impression that the site was created for user benefit.

Before initiating the development of the website, a few drawings of potential layouts were created. The purpose of this was to determine how users wanted to answer questions on the web. The drawings were designed to make it easier for the individual asked questions to understand the concept of each way to display the questions. One drawing represented a list of questions to effectively answer all questions quickly. The second option was designed in a similar way, with the difference being the addition of an introduction to explain the purposes of the questions. The third and final drawing began with the display of an introduction, and each question was presented to the user individually. Lastly, a summary arranged all questions on the screen, and the user was given the possibility to go back and change their answers if so desired. These drawings are displayed in Figure 3.6.

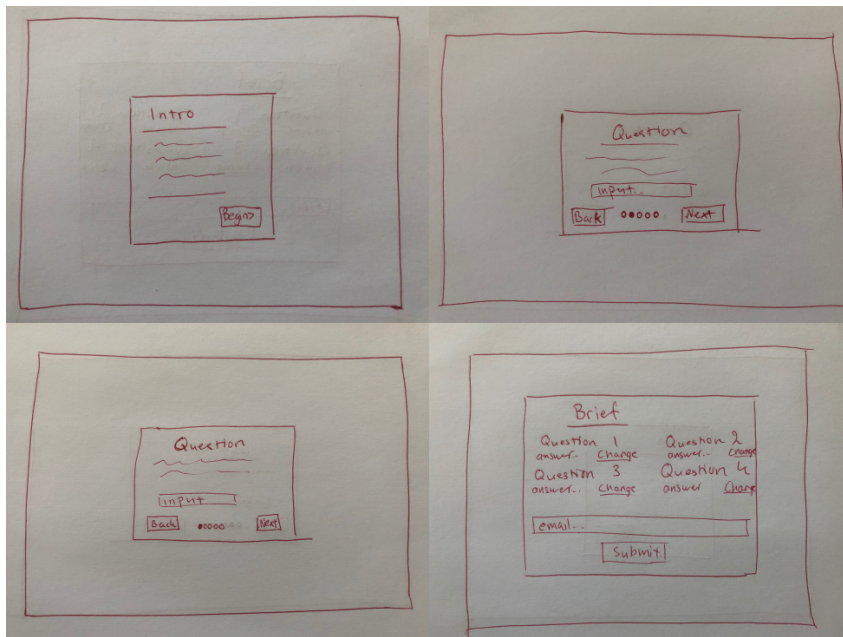


Figure 3.6: Image of the drawings that people preferred when asked how they would want to be asked questions on the web.

3.3 User Testing the Web Survey

The purpose of the test was not to identify the gambling problem of the people testing the website, but rather to test if the system worked as anticipated and to optimize the user experience of the website. As a result, users were given the freedom to answer whatever they wanted on all the questions, instead of giving information about their own gambling habits. Upon submission of all questions and after providing an email address, the subject would receive an email informing them about what results their answers generated. The subjects were then asked if they felt like the application was credible. The number of participants were five. This was due to an article written by Norman[42], where he explains why only five users needs to be tested. The reason is because as soon as the first user has been tested, knowledge about almost one third of the usability of the design is known. Each new user will provide new information about the usability but they will most likely mention more of the same as previous users did. At subject number five, the amount of new knowledge collected versus time spent on doing tests is not considered worth it. Due to ethical reasons, random people in the corridors of Umeå University, instead of actual problem gamblers,

3.3. User Testing the Web Survey

were asked to conduct the test. This survey was not conducted to test whether users were problem gamblers. As mentioned in section 3.3, it was rather to make sure that the decision tree could produce a decision and notify depending input data.

4 Result

Each section can be associated with a section from the chapter discussing methodology. First, information gathered from expert interviews will be presented, followed by the programming language, algorithm, and significant features chosen for the project. Additional information will be provided about how the process of building and testing a decision tree was conducted, along with the development of a web-based service, and a potential way of communicating with problem gamblers about their issues.

4.1 Interviews

The first expert interviewed was a former problem gambler that now works to help others overcome similar problems. As anticipated, when informing the interviewee that this thesis was being written by a founder of a gambling platform, the individual was hesitant to participate. However, after mentioning that the platform solely offers skill-based games, the participant was piqued by interest and agreed to answer some questions, but requested anonymity. For the purposes of this discussion, the individual will be referred to as Alfred. Alfred answered the question exploring the significant features a problem gambler has. He stated that the two main characteristics of a problem gambler include spending a large number of deposits when out of money and devoting a copious amount of time to gambling. He also mentioned that many people engage in destructive gambling when under the influence of alcohol. In order to identify this kind of behavior, one might compare the behavior of users gambling on weekdays with the behavior of users gambling on the weekends, specifically in the evenings. This means that two possible features include day of the week and time of day. Alfred also said that features differ between users with gambling disorders. Some might win money, but they spend almost all their waking time gambling. Others might play zero to a few hours per day and lose everything they deposit each time. Another factor to consider, per Alfred, is player salary; therefore, it might be more efficient to examine the ratio between the user's current account balance and deposited money.

$$(CurrentAccountBalance + WithdrawnMoney) / DepositedMoney \quad (4.1)$$

Given that someone has an account balance of 200, has withdrawn 400 and deposited 1000, this someone would have a ratio of 0,6.

$$(200 + 400) / 1000 = 0,6 \quad (4.2)$$

Having a ratio of 0,6 is equivalent to having lost 40% of the amount of deposited money. Making these calculations opens up the possibility to identify gamblers no matter what their salaries are.

Another person that was contacted was found through posting the survey on reddit. He explained that the absolute biggest problem for him was that it is easy to forget the feeling of losing a lot of money. He explained it as a loop where every time he had lost a big

amount he felt unhappy to the extent that he was sure that he would never gamble again. Nevertheless, without any other action, it was not long until he forgot that pain and got back to gambling again. It took him a lot of time to finally get to a spot where he was able to continue to associate future gambling with the pain from past losses.

A third person being interviewed explained that he never had a problem with money because he had a lot of it. His biggest problem was therefore spending too much time chasing losses, preventing him from spending time doing things he loved. He stated that he always said to himself to play for a certain amount, but as soon he reached zero he deposited again and again trying to catch up on the losses he made, making bigger and bigger bets each time. He said that spending much time defiantly is the biggest issue amongst rich problem gamblers.

4.2 Survey

This section shows the result obtained from the survey. 22 people with prior knowledge about problem gambling took the survey and 13 of these were, or had been gambling addicts in the past. The most significant behavioral feature, with 77,3%, was spending a lot of money. On second place, with 18,2%, came the feature of spending most of the time awake gambling, see figure 4.1.

Survey Answers

Using Machine Learning to Identify Potential Problem Gamblers

22 answers

What type of behavior would you say is the most critical for a problem gambler?

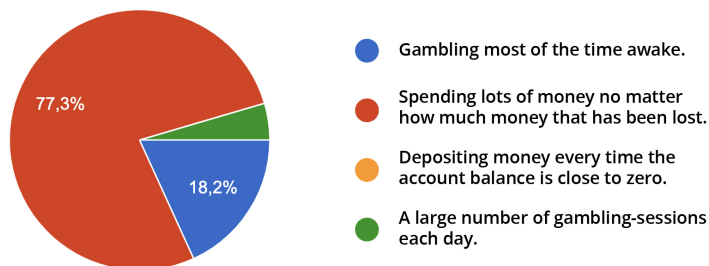


Figure 4.1: The answers of this question gives a convincing hint that the most significant complication with problem gamblers is their urge to spend lots of money. More specifically, their obsession of trying to chase prior losses.

The last four questions were answered solely by the current and prior gambling addicts. The result shows that almost everyone would be comfortable with getting notifications, giving them a hint about their current risk of developing a problem, depending on their

4.2. Survey

behavioral patterns. It also shows that some would reduce their time gambling if they felt like gambling organizations actually cared about their well being. Most of the people would feel okay with being banned temporarily because of them gambling too much. Also, about half of the respondents would actually seek help if they got a notification telling them why they should get support and where to reach out to get help, see figure 4.2.

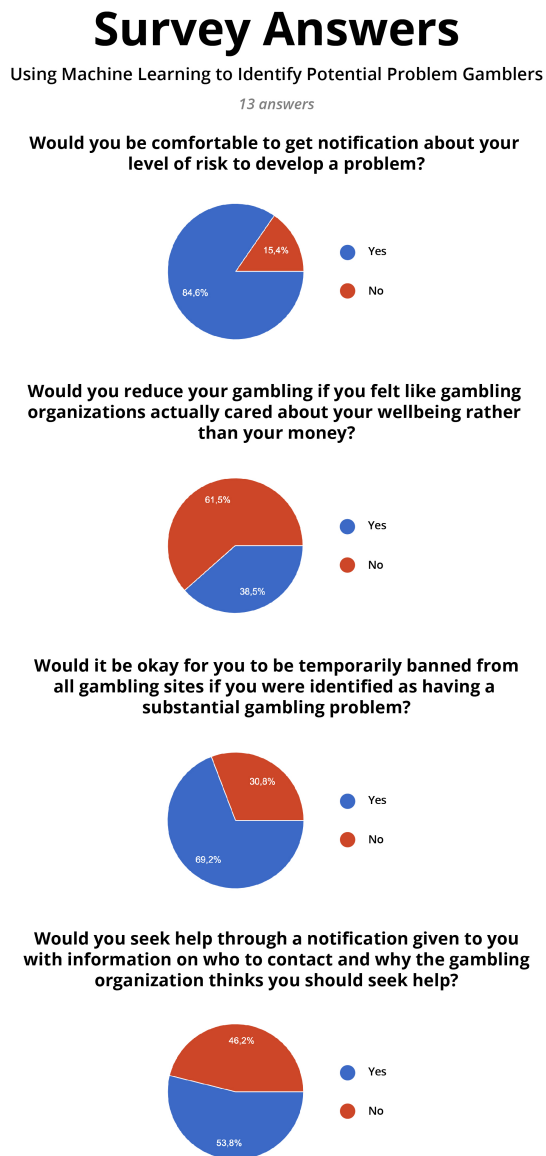


Figure 4.2: These questions shows that most of problem gamblers would want to get help if they felt like they had the chance.

The result shows that the majority of people with gambling problems would not be offended by getting notifications or about organizations collecting data about their gambling habits. Furthermore, the majority would not be unfamiliar with getting information on where to reach out to get help through notifications, and would actually seek help if they got one.

4.3 Choosing Significant Features and Labels

After reviewing the interviews presented in section 4.1 and the literature study[39], multiple features could be considered. Amongst these were that problem gamblers:

- spends a lot of time gambling.
- returns to gambling to chase previous losses.
- hopes to get the ‘big win’.
- refuses to explain behaviour or lies about it.
- prefers to gamble over other activities.
- repeatedly seeks activities that produce a ‘high’.

Because the features needed to be interpreted by computers they had to be converted into something that can be measurable by computers. This meant that a few of the stated features were ruled out, and the remaining once that were used to produce the decision tree were:

- Time spent gambling.
- The number of deposit made when account balance equals zero.
- The ratio from Equation 4.1.

This means that while the decision tree algorithm is gathering three features, five features needs to be gathered from the users. This is because the ratio is calculated through current account balance, withdrawn money and deposited money. The three listed features were partly chosen because experts had said that these were part of the most significant once, but also because the most accurate answers when testing, came from a decision tree built on those features.

4.4 Building the Decision Tree

After 26 different unsuccessful efforts to build a tree with desired results, the final tree was built using 24,286 fictive gamblers, see function; `build_and_get_tree`, figure ?? and ?? in appendix. The entire process took 18 minutes and 38 seconds to complete, and approximately 1,080 questions were asked for classification, see figure ?? in appendix. This means that it took approximately eight hours to finally get the final build done. Taking into account that actions had to be made every time the tree failed to reach a certainty of 95% it took a full week to just come up with the correct training data.

4.5 Testing the Decision Tree

Every time the tree was built, a test was automatically conducted at the end of the building process, see function test, figure ?? and ?? in appendix. Each time the test failed to generate enough correct answers, the training data was modified. The testing data was closely examined in order to determine where the system guessed incorrectly. This process clarified where to provide more information to give the decision tree a higher chance of computing the correct answer. The information was modified until the decision tree could produce the correct answers on 95% of the given testing data. See Figure 4.3 to see the data that the decision tree was presented with. In order to achieve 95% accuracy, 23 out of 24 guesses were required to be correct. An example of how the decision tree looked in the early processes of training is illustrated in Figure 4.4. On the last line of the test, the decision tree predicted the value of one when it should have guessed five. This discrepancy was due to human error when creating the data. The error was due to a simple miss-click when creating the training data, typing 1.1 instead of 0.1 in the win/loss ratio feature. After completing the training 26 times, it yielded 100% accuracy on the testing data (see Figure 4.5 for reference). The training was then conducted five additional times with the same test to ensure that significant exceptions were not occurring by mistake. The number of tests conducted amounted to 144, and the test failed five times, which resulted in a 96.53% chance of success. None of the five failed samples had an error larger than one from the predicted value.

Time Spent (h)	Win/Loss Rate	Deposits when out of Money	Risk Level
1	1	1	0
6	5	1	0
9	9	2	0
3	0.9	3	0
4	1	4	1
1	0.4	2	1
8	1.1	1	1
7	1.05	2	1
4	0.5	2	2
3	0.6	3	2
6	0.45	4	2
10	2	0	2
4	0.4	6	3
9	0.55	7	3
7	1.3	5	3
8	0.61	5	3
12	2	10	4
3	0.3	7	4
7	0.32	7	4
5	0.4	9	4
12	0.3	20	5
1	0	30	5
17	1.3	2	5
0	1	34	5
Chance of Success: 100%			

Figure 4.3: A list of testing data that was used at the end of each tree-building experiment. The data provided for building the tree was rewritten until it could guess with 95% accuracy on this list. The information used to create this list was gathered with the help of expert interviews.

4.5. Testing the Decision Tree

Test #	Actual Result	Predicted Result	Is Correct
1	0	0	Yes
2	0	0	Yes
3	0	3	No
4	0	1	No
5	1	1	Yes
6	1	2	No
7	1	1	Yes
8	1	1	Yes
9	2	2	Yes
10	2	2	Yes
11	2	2	Yes
12	2	2	Yes
13	3	3	Yes
14	3	3	Yes
15	3	3	Yes
16	3	3	Yes
17	4	4	Yes
18	4	3	No
19	4	4	Yes
20	4	4	Yes
21	5	5	Yes
22	5	5	Yes
23	5	5	Yes
24	5	1	No
Chance of Success: 79.17%			

Figure 4.4: Shows the fifteenth test, which resulted in a 79.17% chance of success.

Test #	Actual Result	Predicted Result	Is Correct
1	0	0	Yes
2	0	0	Yes
3	0	0	Yes
4	0	0	Yes
5	1	1	Yes
6	1	1	Yes
7	1	1	Yes
8	1	1	Yes
9	2	2	Yes
10	2	2	Yes
11	2	2	Yes
12	2	2	Yes
13	3	3	Yes
14	3	3	Yes
15	3	3	Yes
16	3	3	Yes
17	4	4	Yes
18	4	4	Yes
19	4	4	Yes
20	4	4	Yes
21	5	5	Yes
22	5	5	Yes
23	5	5	Yes
24	5	5	Yes
Chance of Success: 100%			

Figure 4.5: Shows the first of the six final tests, which amounted to a 100% chance of success

4.6 Final Result of the Developed Web Survey

Figures 4.6 through 4.12 show the resultant questions from the survey and the design produced by using the drawings shown in Figure 3.6. All of the questions can be seen in Figure 4.10, and the answers are presented in the email in Figure 4.12.

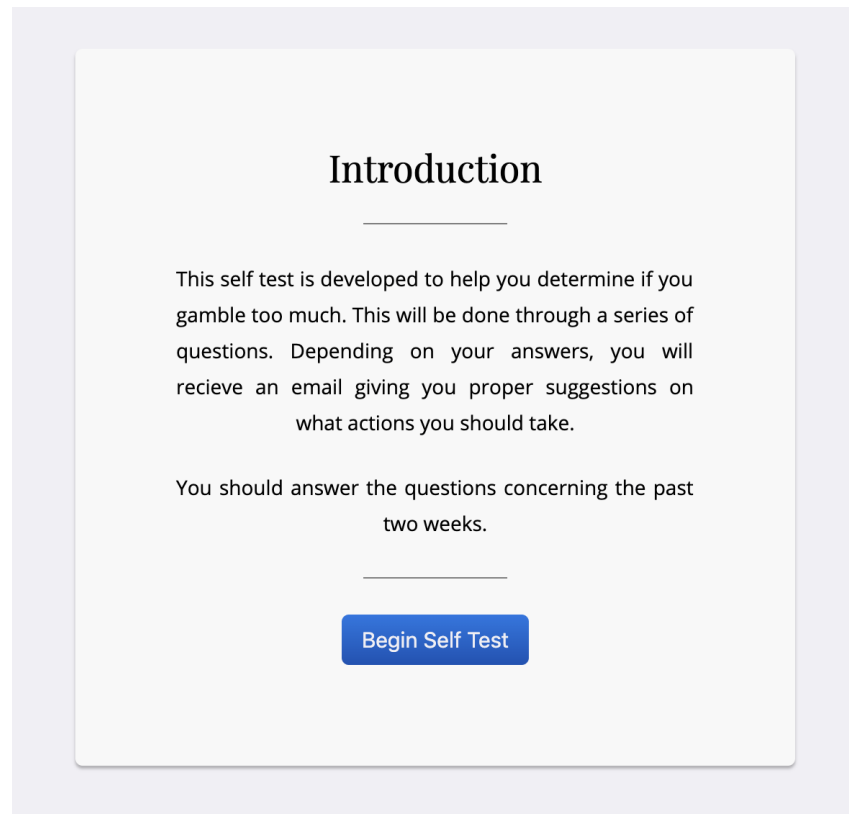


Figure 4.6: This is the first page of the self-identification. Users are introduced to how the test will be carried out, and they are presented with an idea about answering the questions, keeping in mind that they should base their answers on the past two weeks of their lives.

How many hours per day have you been gambling?

0 24

4 hours

Back Next

Question 1/5

The image shows a user interface for a survey question. The question is "How many hours per day have you been gambling?". Below the question is a horizontal range input slider. The slider has a blue circular handle positioned at the 4-hour mark. The range is labeled from 0 to 24. Below the slider, there are two blue buttons labeled "Back" and "Next". At the bottom of the interface, it says "Question 1/5".

Figure 4.7: To limit the number of possible inputs a user could make, a range input was implemented for this question.

How much money have you deposited?

Your answer

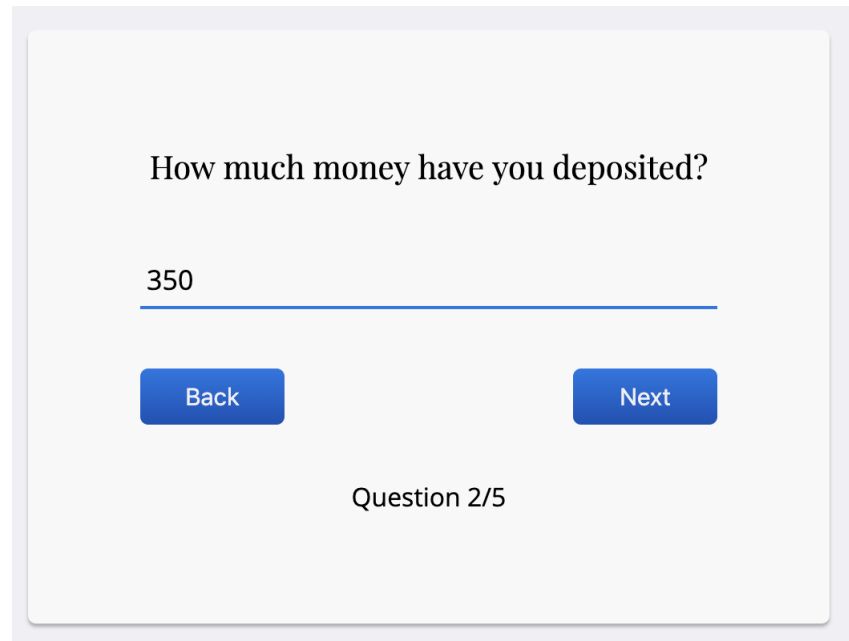
Back Next

Question 2/5

The image shows a user interface for a survey question. The question is "How much money have you deposited?". Below the question is a text input field with the placeholder text "Your answer". Below the input field, there are two blue buttons labeled "Back" and "Next". At the bottom of the interface, it says "Question 2/5".

Figure 4.8: To be able to proceed to the next question, an input has to be entered. This is shown in the form of the transparent “next” button.

4.6. Final Result of the Developed Web Survey



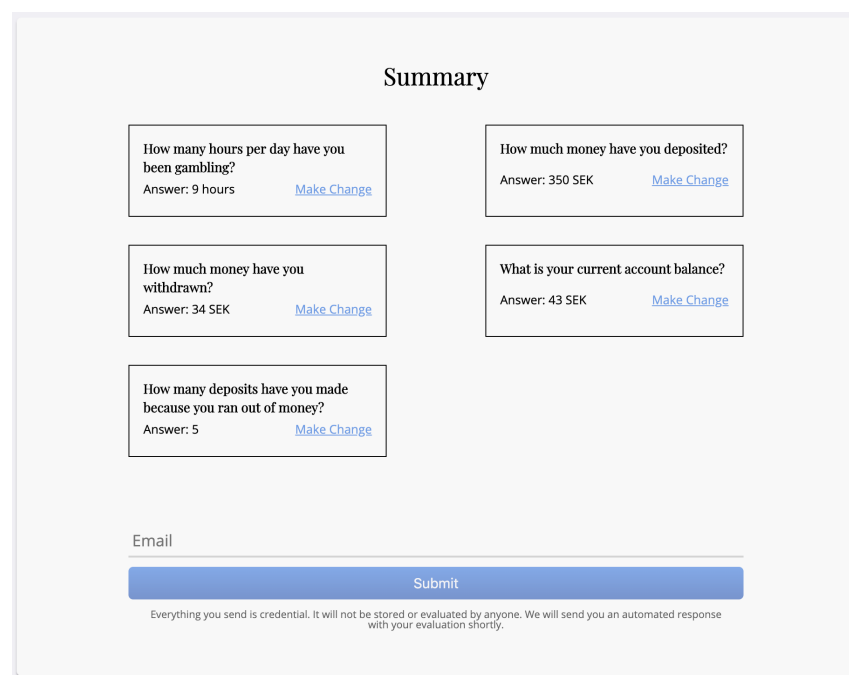
How much money have you deposited?

350

Back Next

Question 2/5

Figure 4.9: This shows when a user has entered an input. It is only possible to enter numbers with the format of integers or decimals.



Summary

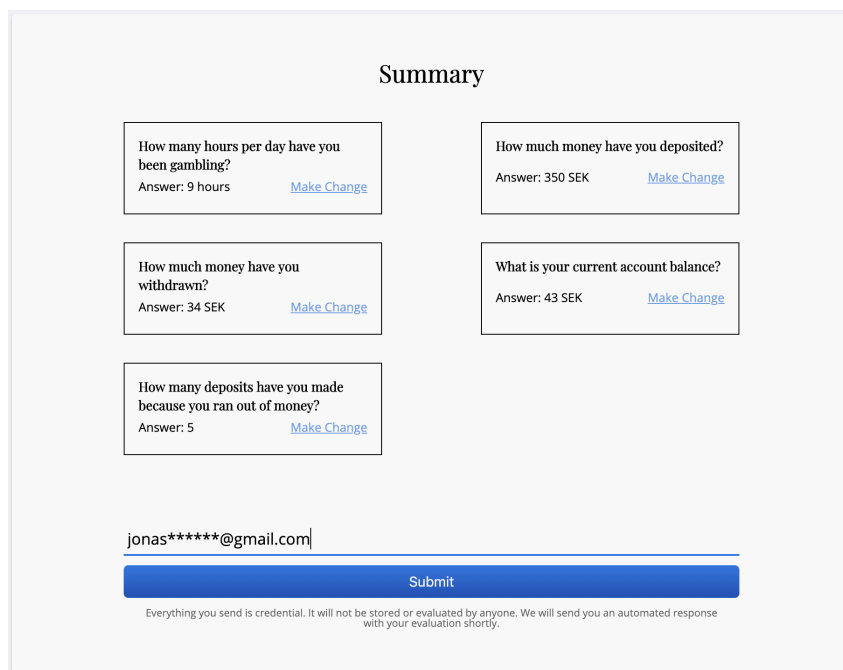
How many hours per day have you been gambling? Answer: 9 hours Make Change	How much money have you deposited? Answer: 350 SEK Make Change
How much money have you withdrawn? Answer: 34 SEK Make Change	What is your current account balance? Answer: 43 SEK Make Change
How many deposits have you made because you ran out of money? Answer: 5 Make Change	

Email

Submit

Everything you send is credential. It will not be stored or evaluated by anyone. We will send you an automated response with your evaluation shortly.

Figure 4.10: The summary of all questions is shown above. Here, it is possible to make changes to whatever questions need modification. The email input has been validated so that nothing more than email addresses can be sent to the API.



Summary

How many hours per day have you been gambling?
Answer: 9 hours [Make Change](#)

How much money have you deposited?
Answer: 350 SEK [Make Change](#)

How much money have you withdrawn?
Answer: 34 SEK [Make Change](#)

What is your current account balance?
Answer: 43 SEK [Make Change](#)

How many deposits have you made because you ran out of money?
Answer: 5 [Make Change](#)

jonas*****@gmail.com

Submit

Everything you send is credential. It will not be stored or evaluated by anyone. We will send you an automated response with your evaluation shortly.

Figure 4.11: When a valid email address has been entered into the input field, it is possible for the user to submit, as shown above.

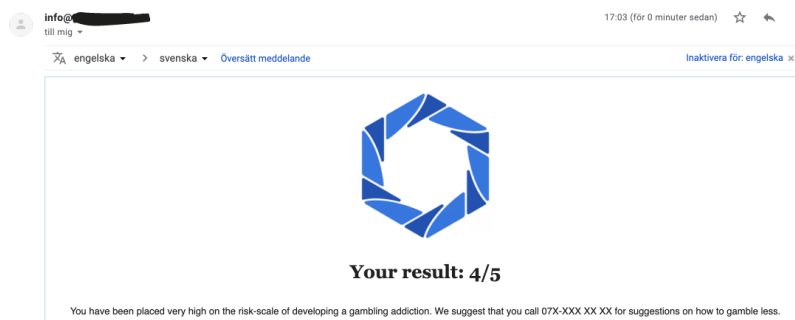


Figure 4.12: After the form has been submitted, the decision tree will make a decision based on the level of risk the user has of developing a gambling addiction. A proper suggestion on what actions to take will then be sent to that user via email.

4.7 User Testing

None of the subjects answered that they found the website to be unconvincing. Furthermore, there were no complications for the subjects when completing the questions; they understood the step-by-step process of answering the questions. However, the subjects revealed that they thought the information provided in the post-test email needed further development, see figure 4.12. The email implementation was created with python, see figure ??, ?? and ?? in appendix.

5 Discussion

This chapter will include why certain decisions were made and what could have been done differently if the thesis were to be conducted again. The specific topics discussed are the expert interviews, the choice of features, building and testing the decision tree, and user testing.

5.1 The Impact a Service Like This Could Have

Even if there were no more than 13 persons answering the survey regarding their opinion on getting notification and-or help through a system like the one that was developed during this thesis, it is still interesting to see that the majority would seek help if presented with how to proceed in the matter. According to Genius Health [43], in America only, there are about 2.5 million adults suffering from compulsive gambling, 3 million are considered problem gamblers and around 15 million adult are under the risk of becoming problem gamblers. To put the data gathered from the survey into perspective, this would mean that approximately 11 out of 20,5 million of these Americans would seek help if they got a message about them gambling too much and information about how to get help.

A problem when trying to sell the idea to gambling organizations is that they will be losing a lot of their customers, due to people not gambling anymore. Therefore, a system like this would need to be required by authorities for gambling companies to use it. Due to this, the main stakeholders would become national authorities instead of gambling organizations.

5.2 Interviews

When looking to find experts to interview, a thorough purpose statement about why the thesis was being conducted would likely have been beneficial and facilitated cooperation amongst the participants. Because most of the individuals contacted declined to answer any questions, this resulted in progress limitations. A solution around this problem might have been to develop several backup plans, even though the occurrence of this situation was not anticipated. The development of multiple plans might have prevented the cessation of progress experienced as a result of the refusals to participate. Additionally, the literature study conducted would have been more of a substantial benefit if the research had been confirmed by a larger amount of experts.

5.3 Building and Testing the Decision Tree

The decision tree was built with approximately 26,000 fictive users per training. This took about 15-20 minutes each time it was built. If this process had been more time efficient, more elements could have been incorporated, which would have led to more combinations

of data. With more combinations of data, more labels could be used. These labels could include the reason for why someone achieves a specific result. For example, if an individual gambles twelve hours a day without losing anything, that player's problem is probably not that they are impulsively betting money, but that they are spending too much time gambling. Testing the decision tree could have been conducted with more testing data. The reason for not doing this was simply because the results were already of high quality without more data. With the decision tree guessing the correct output 96% of the times without an error greater than one, it was decided that further testing would not be necessary. Furthermore, the original plan for the study was to test the system on actual problem gamblers, but no organization that was contacted was willing to participate.

As mentioned previously in the methods chapter, a greater number of features could have benefited this study. The reason this was not performed was because it would have to involve users to submit their salaries, the total amount of money they own, and more. The request for this information would verge on impeding user integrity. Another feature that could have been utilized was to directly ask users if they felt they had a gambling problem, which might have led to dishonesty and denial, which could exacerbate potential gambling problems and their associated consequences.

Pruning is more often than not crucial when it comes to building decision trees with CART. This is due to the tree generating very specific classifications. What this means is that its goal is to try and have only one element per leaf node, which in turn makes it very hard for the system to classify data that has never been seen before. This is mostly a problem when using questions that require something to have one specific value. Because the system implemented through this thesis only use operations that check if something is greater than something else instead of checking if something is equal to something else, this is not a problem. In other words, the system will have leaf nodes that inhabit multiple elements each without using pruning.

As mentioned in the theory section, about half of all problem gamblers have a mood disorder. Because of that, this could have been a crucial part of the thesis. Nevertheless, because of the small number of problem gamblers contacted, due to organizations not wanting to participate in helping with the thesis, mood disorders were not considered when conducting this essay. Because such a big number of the problem gambling population have a mood disorder, the typical behavioral patterns of such persons could have been studied further. The outcome of such data could possibly have resulted in an increased amount of behavioral patterns to include into the decision tree.

6 Conclusion

This thesis aimed to answer if it would be possible to identify problem gamblers using machine learning. Furthermore, how would people react to being monitored by such a system and if identification would be possible, would it actually be able to help people to get out of their bad habits? The process of conducting this thesis helped to come to the conclusion that machine learning, if implemented according to specific needs, is able to accomplish almost anything. After all, humans are based on inputs and outputs, which makes it possible to convert those outputs to patterns that computers can understand. Therefore, the behavioral patterns that problem gamblers possess, although there are many, can be used for identification. Nevertheless, it is important to take into account the ethical questions emerging when collecting data about people. Would they feel that it is okay to gather information about their problems. The majority of people conducting the survey, specifically sent out to problem gamblers, gave the impression that they would accept a system that is created to help them. This led to the belief that, if conducted right, machine learning could be used to create a business in the area of problem gambling in the future.

6.1 Future Work

6.1.1 Storing Data About Users Behaviour

To be able to integrate the API to a corporation's gambling platform, it would need to be able to gather information about users from a database. This could, of course, raise a concern for the corporation using the API. However, a more efficient way of notifying users would be for the API to be able to gather real-time information about users. This would enable it to notify immediately when a user exhibits an addictive behavior. When the API is utilized today, the corporation using it would need to decide when and how often to make a request to identify a potential gambling problem. This means that it would not be fully automated whilst connected to the gambling service. To enable that automation, push events would need to be implemented through web sockets, giving the API/Server the possibility to directly communicate to end users without the gambling platform client needing to send request to notify users.

6.1.2 Making Decisions for Problem Gamblers

Although lots of development would be required for the system to fully work, a system that actually takes on other user behavior could be implemented. This means that if a user obtains the highest possible score on the scale, they could be automatically suspended from gambling. To make a system like this work, all gambling platforms would need to cooperate. Collaborations such as this would mean that the gambling companies assume responsibility to connect the AP so that the API can disable gambling on all platforms. Since the beginning of 2019 in Sweden, all Swedish gambling companies with a gambling license need to connect to "spelpaus.nu." Spelpaus enables the possibility for users to suspend themselves

from all Swedish gambling companies. Working with Spelpaus could possibly enable implementing this system in Sweden, which would be a great start.

6.1.3 Giving Gambling Companies a tag for Responsibility

This system will have potential to grow. It may reveal that corporations using the system receive a tag similar to the AAA tag, meaning that gambling companies would want to use a similar system to make them appear to be a serious and trustworthy corporation. This would be an addition to spelpaus.nu, but for gambling organizations all over the world instead of only in Sweden.

6.1.4 Open Source

Because Source Empire ordered this project in goodwill. This API will be available to all with no licensing required except mentioning that Source Empire is the company implementing the system. The hope is that people from all over the world will make improvements and come up with ideas on how to develop the system for different use cases, but with the purpose always being to help problem gamblers get out of their bad habits.

References

- [1] Altha Stewart. “Heading Into the Home Stretch”. In: *Psychiatric News* 54.3 (Feb. 2019), appi.pn.2019.2a22. ISSN: 0033-2704. DOI: 10.1176/appi.pn.2019.2a22. URL: <https://psychiatryonline.org/doi/10.1176/appi.pn.2019.2a22>.
- [2] Robert F Leeman and Marc N and Potenza. “Similarities and differences between pathological gambling and substance use disorders: a focus on impulsivity and compulsivity”. In: *Psychopharmacology* 219.2 (2012), pp. 469–490. ISSN: 0033-3158. DOI: 10.1007/s00213-011-2550-7. URL: <http://10.0.3.239/s00213-011-2550-7%20https://dx.doi.org/10.1007/s00213-011-2550-7>.
- [3] Wolfram and Schultz. “Predictive Reward Signal of Dopamine Neurons”. In: *Journal of Neurophysiology* 80.1 (1998), pp. 1–27. ISSN: 0022-3077. DOI: 10.1152/jn.1998.80.1.1. URL: <http://10.0.4.128/jn.1998.80.1.1%20https://dx.doi.org/10.1152/jn.1998.80.1.1>.
- [4] *The Role of Dopamine in Gambling Withdrawal*. URL: <https://www.algamus.org/blog/role-of-dopamine-in-gambling-withdrawal> (visited on 02/28/2019).
- [5] *The History of Gambling - Complete Gambling History Timeline*. URL: <https://www.gambling.net/history/> (visited on 09/30/2019).
- [6] *History of Gambling 2019 - Early Beginnings To The Future*. URL: <https://www.onlinegambling.com/history/> (visited on 09/30/2019).
- [7] *The History of Gambling*. URL: <http://www.gypsyware.com/gamblingHistory.html> (visited on 09/30/2019).
- [8] *A Brief History of Gambling – edgfund – Medium*. URL: <https://medium.com/edgfund/a-brief-history-of-gambling-a7f46dbf4403> (visited on 02/13/2019).
- [9] Mark Griffiths and Adrian Parke. “Internet Gambling”. In: *Encyclopedia of Internet Technologies and Applications*. IGI Global, Jan. 2008, pp. 228–234. DOI: 10.4018/978-1-59140-993-9.ch033. URL: <http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-59140-993-9.ch033>.
- [10] Jon E. Grant et al. “Pathological gambling and mood disorders: Clinical associations and treatment implications”. In: *Journal of Affective Disorders* 92.1 (May 2006), pp. 109–116. ISSN: 0165-0327. DOI: 10.1016/J.JAD.2005.12.040. URL: <https://www.sciencedirect.com/science/article/pii/S0165032705004106>.
- [11] Wayne C and Drevets. “Neuroimaging studies of mood disorders”. In: *Biological Psychiatry* 48.8 (2000), pp. 813–829. ISSN: 0006-3223. DOI: 10.1016/s0006-3223(00)01020-9. URL: [http://10.0.3.248/s0006-3223\(00\)01020-9%20https://dx.doi.org/10.1016/s0006-3223\(00\)01020-9](http://10.0.3.248/s0006-3223(00)01020-9%20https://dx.doi.org/10.1016/s0006-3223(00)01020-9).

- [12] Sally Monaghan. “Responsible gambling strategies for Internet gambling: The theoretical and empirical base of using pop-up messages to encourage self-awareness”. In: *Computers in Human Behavior* 25.1 (Jan. 2009), pp. 202–207. ISSN: 0747-5632. DOI: 10.1016/J.CHB.2008.08.008. URL: <https://www.sciencedirect.com/science/article/pii/S0747563208001659>.
- [13] Martin Cloutier, Robert Ladouceur, and Serge and Sévigny. “Responsible Gambling Tools: Pop-Up Messages and Pauses on Video Lottery Terminals”. In: *The Journal of Psychology* 140.5 (2006), pp. 434–438. ISSN: 0022-3980. DOI: 10.3200/jrlp.140.5.434-438. URL: <http://10.0.12.128/jrlp.140.5.434-438%20https://dx.doi.org/10.3200/JRLP.140.5.434-438>.
- [14] *Brawl Gaming*. URL: <https://brawlgaming.com/> (visited on 02/19/2019).
- [15] *What is Artificial Intelligence (AI)? - Definition from Techopedia*. URL: <https://www.techopedia.com/definition/190/artificial-intelligence-ai> (visited on 02/20/2019).
- [16] John R. Koza et al. “Automated Design of Both the Topology and Sizing of Analog Electrical Circuits Using Genetic Programming”. In: *Artificial Intelligence in Design '96*. Dordrecht: Springer Netherlands, 1996, pp. 151–170. DOI: 10.1007/978-94-009-0279-4_9. URL: http://www.springerlink.com/index/10.1007/978-94-009-0279-4%7B%5C_%7D9.
- [17] *Supervised Machine Learning: Classification – Towards Data Science*. URL: <https://towardsdatascience.com/supervised-machine-learning-classification-5e685fe18a6d> (visited on 03/03/2019).
- [18] *What is supervised learning ? - Definition from WhatIs.com*. URL: <https://searchenterpriseai.techtarget.com/definition/supervised-learning> (visited on 03/03/2019).
- [19] *An introduction to Reinforcement Learning – freeCodeCamp.org*. URL: <https://medium.freecodecamp.org/an-introduction-to-reinforcement-learning-4339519de419> (visited on 05/20/2019).
- [20] *Introduction to Classification & Regression Trees (CART) - Data Science Central*. URL: <https://www.datasciencecentral.com/profiles/blogs/introduction-to-classification-regression-trees-cart> (visited on 03/03/2019).
- [21] Wei-Yin Loh. “Classification and regression trees CLASSIFICATION TREES”. In: *C 1* (2011), pp. 14–23. DOI: 10.1002/widm.8. URL: <https://www.stat.wisc.edu/%7B~%7Dloh/treeprogs/guide/wires11.pdf>.
- [22] *Overfitting in Machine Learning: What It Is and How to Prevent It*. URL: <https://elitedatascience.com/overfitting-in-machine-learning> (visited on 03/07/2019).
- [23] *Machine Learning: Pruning Decision Trees — Displayr*. URL: <https://www.displayr.com/machine-learning-pruning-decision-trees/> (visited on 03/07/2019).
- [24] Mark A Hall. *Correlation-based Feature Selection for Machine Learning*. Tech. rep. 1999. URL: <https://www.lri.fr/%7B~%7Dpierres/donn%7B%5C%7DE9es/save/these/articles/lpr-queue/hall99correlationbased.pdf>.

References

- [25] Seyyed Salman Alavi et al. “Behavioral Addiction versus Substance Addiction: Correspondence of Psychiatric and Psychological Views.” In: *International journal of preventive medicine* 3.4 (Apr. 2012), pp. 290–4. ISSN: 2008-8213. URL: <http://www.ncbi.nlm.nih.gov/pubmed/22624087><http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3354400>.
- [26] “Systems/Circuits Parceling Human Accumbens into Putative Core and Shell Dis-sociates Encoding of Values for Reward and Pain”. In: (2013). DOI: 10.1523/JNEUROSCI.1731-13.2013. URL: <http://www.jneurosci.org/content/jneuro/33/41/16383.full.pdf>.
- [27] “Prefrontal/accumbal catecholamine system processes high motivational salience.” In: *Frontiers in behavioral neuroscience* 6 (2012), p. 31. ISSN: 1662-5153. DOI: 10.3389/fnbeh.2012.00031. URL: <http://www.ncbi.nlm.nih.gov/pubmed/22754514><http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3384081>.
- [28] *Gambling Disorder (Compulsive Gambling, Pathological Gambling) — Psychology Today*. URL: <https://www.psychologytoday.com/us/conditions/gambling-disorder-compulsive-gambling-pathological-gambling> (visited on 03/07/2019).
- [29] *Bipolar Disorder and Gambling Addiction - Bipolar Disorder Center - Everyday Health*. URL: <https://www.everydayhealth.com/bipolar-disorder/can-bipolar-disorder-lead-to-gambling-addiction.aspx> (visited on 02/27/2019).
- [30] *The Self Medication Theory of Addiction*. URL: <https://www.verywellmind.com/the-self-medication-theory-of-addiction-21933> (visited on 02/27/2019).
- [31] *Google Scholar*. URL: <https://scholar.google.se/> (visited on 03/18/2019).
- [32] *Python vs. R: Which Should You Choose For Your Next ML Project? - DZone AI*. URL: <https://dzone.com/articles/python-or-r-which-should-you-choose-for-your-next> (visited on 05/21/2019).
- [33] Bhumika Gupta, Pauri Uttarakhand, and India Aditya Rawat. *Analysis of Various Decision Tree Algorithms for Classification in Data Mining*. Tech. rep. 8. 2017, pp. 975–8887. URL: <https://pdfs.semanticscholar.org/fd39/e1fa85e5b3fd2b0d000230f6f8bc9dc6.pdf>.
- [34] *An Introduction to Decision Tree Learning: ID3 Algorithm*. URL: <https://medium.com/machine-learning-guy/an-introduction-to-decision-tree-learning-id3-algorithm-54c74eb2ad55> (visited on 05/24/2019).
- [35] *What is the C4.5 algorithm and how does it work? – Towards Data Science*. URL: <https://towardsdatascience.com/what-is-the-c4-5-algorithm-and-how-does-it-work-2b971a9e7db0> (visited on 05/24/2019).
- [36] *An Introduction to Random Forest – Towards Data Science*. URL: <https://towardsdatascience.com/random-forest-3a55c3aca46d> (visited on 05/24/2019).
- [37] *(16) Let’s Write a Decision Tree Classifier from Scratch - Machine Learning Recipes #8 - YouTube*. URL: https://www.youtube.com/watch?v=d12ra3b%7B%5C_%7DM-0 (visited on 04/13/2019).
- [38] *Decision tree google dev code*. URL: https://github.com/random-forests/tutorials/blob/master/decision_tree.ipynb (visited on 04/13/2019).

- [39] Carmen Messerlian, Meredith Gillespie, and Jeffrey L Derevensky. “Beyond drugs and alcohol: Including gambling in a high-risk behavioural framework.” In: *Paediatrics & child health* 12.3 (Mar. 2007), pp. 199–204. ISSN: 1205-7088. DOI: 10.1093/pch/12.3.199. URL: <http://www.ncbi.nlm.nih.gov/pubmed/19030359><http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC2528700>.
- [40] *What is an API Endpoint? — SmartBear Software Resources*. URL: <https://smartbear.com/learn/performance-monitoring/api-endpoints/> (visited on 04/14/2019).
- [41] *Color Wheel Pro: Color Meaning*. URL: <http://www.color-wheel-pro.com/color-meaning.html> (visited on 04/16/2019).
- [42] *Why You Only Need to Test with 5 Users*. URL: <https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/> (visited on 05/10/2019).
- [43] *Gambling Addiction — Gambling Management - Geneus Health*. URL: <https://www.geneushealth.com/gambling-genetic-rehab-texas> (visited on 05/24/2019).

Appendices

A Main

```
File - /Users/jonas/PycharmProjects/ml-identifying-problem-gamblers/main.py
1 from __future__ import print_function
2
3 from abc import ABC
4
5 import demjson as demjson
6
7 from api.rest import Rest
8
9 from decision_tree.tree import Tree
10 from mail.mail import Mail
11
12 from tornado.web import Application, RequestHandler
13 from tornado.ioloop import IOLoop
14
15
16 class MainHandler(RequestHandler, ABC):
17     def get(self):
18         self.write({'succeeded': 'welcome to problem gambling identifier'})
19
20
21 class TreeHandler(RequestHandler, ABC):
22     mail = Mail()
23     tree = Tree()
24     print('\nServer is up and running.')
25
26     def set_default_headers(self):
27         self.set_header("Access-Control-Allow-Origin", "*")
28         self.set_header("Access-Control-Allow-Headers",
29             "Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With")
30         self.set_header('Access-Control-Allow-Methods', 'POST, GET, OPTIONS')
31
32     def get(self):
33         self.write({'message': self.tree.get_actual([12, .3, 20])})
34
35     def post(self):
36         time = demjson.decode(self.request.body)['time']
37         nr_deposits = demjson.decode(self.request.body)['nrdeposits']
38         win_loss = demjson.decode(self.request.body)['win_loss']
39         email = demjson.decode(self.request.body)['email']
40
41         result = self.tree.get_actual([float(time), float(win_loss), int(nr_deposits)])
42
43         self.mail.send_mail(email, result)
44
45         self.write({'succeeded': True})
46
47
48 def make_app():
49     return Application([
50         (Rest.MAIN, MainHandler),
51         (Rest.GET_RATE, TreeHandler)
52     ])
53
54
55 if __name__ == '__main__':
56     app = make_app()
57     app.listen(6060)
58     IOLoop.instance().start()
59
```

Page 1 of 1

Figure A.1

B Tree

```
File - /Users/jonas/PycharmProjects/ml-identifying-problem-gamblers/decision_tree/tree.py
1 import random
2 import sys
3
4 from decision_tree.player import Player
5 from decision_tree.question import Question
6 from decision_tree.leaf import Leaf
7 from decision_tree.decisionnode import DecisionNode
8 from utils import class_counts
9
10 header = ["Time", "Win/Loss Ratio", "Number of Deposits", "Label"]
11
12 # header = ["temperature", "windy", "humidity", "label"]
13
14
15
16 class Tree:
17     tree = None
18
19     def __init__(self):
20         self.nrOfNodes = 0
21         self.builtNodes = 0
22         self.cols = []
23
24         # self.tree = self.build_and_get_golf()
25         # self.tree = self.build_and_get_tree()
26         self.tree = self.build_and_get_tree_small()
27
28         self.print_tree(self.tree)
29         self.test()
30
31     def get(self):
32         return self.tree
33
34     @staticmethod
35     def create_player_list(iteration_span, time_span, win_rate_span, nr_deposit, answer):
36         player_list = []
37         for x in range(random.randint(iteration_span[0], iteration_span[1])):
38             player_list.append(
39                 Player(random.randint(time_span[0], time_span[1]),
40                       random.uniform(win_rate_span[0], win_rate_span[1]),
41                       random.randint(nr_deposit[0], nr_deposit[1]),
42                       answer).get_player_params())
43         return player_list
44
45     @staticmethod
46     def unique_values(rows, col):
47         """Find the unique values for a column in a dataset."""
48         return set([row[col] for row in rows])
49
50     @staticmethod
51     def partition(rows, question):
52         """Partitions a dataset.
53         For each row in the dataset, check if it matches the question. If
54         so, add it to 'true rows', otherwise, add it to 'false rows'."""
55
56         true_rows, false_rows = [], []
57         for row in rows:
58             if question.match(row):
59                 true_rows.append(row)
60             else:
61                 false_rows.append(row)
62         return true_rows, false_rows
63
64     @staticmethod
65     def gini(rows):
66         """Calculate the Gini Impurity for a list of rows.
67         There are a few different ways to do this, I thought this one was
68         the most concise. See:
69         https://en.wikipedia.org/wiki/Decision_tree_learning#Gini_impurity """
```

Page 1 of 6

Figure B.1

File - /Users/jonas/PycharmProjects/ml-identifying-problem-gamblers/decision_tree/tree.py

```

70     counts = class_counts(rows)
71     impurity = 1
72
73     for lbl in counts:
74         prob_of_lbl = counts[lbl] / float(len(rows))
75         impurity -= prob_of_lbl ** 2
76     return impurity
77
78 def info_gain(self, left, right, current_uncertainty):
79     """ Information Gain.
80     The uncertainty of the starting node, minus the weighted impurity of
81     two child nodes. """
82     p = float(len(left)) / (len(left) + len(right))
83     return current_uncertainty - p * self.gini(left) - (1 - p) * self.gini(right)
84
85 def find_best_split(self, rows):
86     """ Find the best question to ask by iterating over every feature / value
87     and calculating the information gain. """
88     best_gain = 0 # keep track of the best information gain
89     best_question = None # keep train of the feature / value that produced it
90     current_uncertainty = self.gini(rows)
91     n_features = len(rows[0]) - 1 # number of columns
92
93     for col in range(n_features): # for each feature
94
95         values = set([row[col] for row in rows]) # unique values in the column
96
97         for val in values: # for each value
98
99             question = Question(col, val, header)
100
101             # try splitting the dataset
102             true_rows, false_rows = self.partition(rows, question)
103
104             # Skip this split if it doesn't divide the
105             # dataset.
106             if len(true_rows) == 0 or len(false_rows) == 0:
107                 continue
108
109             # Calculate the information gain from this split
110             gain = self.info_gain(true_rows, false_rows, current_uncertainty)
111
112             # You actually can use '>' instead of '>=' here
113             # but I wanted the tree to look a certain way for our
114             # toy dataset.
115             if gain >= best_gain:
116                 best_gain, best_question = gain, question
117
118     return best_gain, best_question
119
120 def build_tree(self, rows):
121     """
122     Builds the tree.
123     Rules of recursion:
124     1) Believe that it works.
125     2) Start by checking for the base case (no further information gain).
126     3) Prepare for giant stack traces.
127     """
128
129     # Try partitioning the dataset on each of the unique attribute,
130     # calculate the information gain,
131     # and return the question that produces the highest gain.
132     gain, question = self.find_best_split(rows)
133
134     # Base case: no further info gain
135     # Since we can ask no further questions,
136     # we'll return a leaf.
137
138     if gain == 0:

```

Page 2 of 6

Figure B.2

File - /Users/jonas/PycharmProjects/ml-identifying-problem-gamblers/decision_tree/tree.py

```
139         self.builtNodes += len(rows)
140         self.print_progress()
141         return Leaf(rows)
142
143     # If we reach here, we have found a useful feature / value
144     # to partition on.
145     true_rows, false_rows = self.partition(rows, question)
146
147     # Recursively build the true branch.
148     true_branch = self.build_tree(true_rows)
149
150     # Recursively build the false branch.
151     false_branch = self.build_tree(false_rows)
152
153     # Return a Question node.
154     # This records the best feature / value to ask at this point,
155     # as well as the branches to follow
156     # depending on the answer.
157     return DecisionNode(question, true_branch, false_branch)
158
159 def print_tree(self, node, spacing=""):
160     """World's most elegant tree printing function."""
161
162     # Base case: we've reached a leaf
163     if isinstance(node, Leaf):
164         print(spacing + "Predict", node.predictions)
165         return
166
167     # Print the question at this node
168     print(spacing + str(node.question))
169
170     # Call this function recursively on the true branch
171     print(spacing + '---> True:')
172     self.print_tree(node.true_branch, spacing + " ")
173
174     # Call this function recursively on the false branch
175     print(spacing + '---> False:')
176     self.print_tree(node.false_branch, spacing + " ")
177
178 def classify(self, row, node):
179     """See the 'rules of recursion' above."""
180
181     # Base case: we've reached a leaf
182     if isinstance(node, Leaf):
183         return node.predictions
184
185     # Decide whether to follow the true-branch or the false-branch.
186     # Compare the feature / value stored in the node,
187     # to the example we're considering.
188     if node.question.match(row):
189         return self.classify(row, node.true_branch)
190     else:
191         return self.classify(row, node.false_branch)
192
193 @staticmethod
194 def print_leaf(counts):
195     """A nicer way to print the predictions at a leaf."""
196     total = sum(counts.values()) * 1.0
197     probs = {}
198     for lbl in counts.keys():
199         probs[lbl] = str(int(counts[lbl] / total * 100)) + "%"
200     return probs
201
202 def build_and_get_golf(self):
203     training_data = [
204         ('hot', 'false', 'high', 'Play'),
205         ('cold', 'true', 'low', 'Do not play'),
206         ('hot', 'false', 'normal', 'Play'),
207         ('mild', 'false', 'low', 'Play'),
```

Page 3 of 6

Figure B.3

File - /Users/jonas/PycharmProjects/ml-identifying-problem-gamblers/decision_tree/tree.py

```

208         ('mild', 'true', 'low', 'Do not play'),
209         ('mild', 'true', 'high', 'Do not play'),
210         ('hot', 'true', 'high', 'Play'),
211         ('cold', 'false', 'high', 'Do not play'),
212         ('cold', 'false', 'low', 'Play'),
213         ('hot', 'true', 'normal', 'Play'),
214         ('cold', 'false', 'normal', 'Play'),
215         ('mild', 'true', 'high', 'Do not play'),
216         ('hot', 'true', 'normal', 'Play'),
217         ('cold', 'false', 'normal', 'Play'),
218         ('cold', 'false', 'high', 'Do not play'),
219         ('mild', 'true', 'high', 'Do not play'),
220         ('hot', 'false', 'low', 'Play'),
221         ('hot', 'false', 'normal', 'Play'),
222         ('cold', 'true', 'low', 'Do not play'),
223     ]
224
225     self.nrofNodes = len(training_data)
226     self.print_progress()
227
228     return self.build_tree(training_data)
229
230
231
232     """
233     create_player_list
234     { Parameters: -> 1: number of iterations, 2: hours gambling,
235                   3: win/loss rate, 4: number of deposits, 5: label
236     }
237     """
238
239     def build_and_get_tree(self):
240         training_data = \
241             self.create_player_list([400, 1000], [0, 3], [.8, 3], [0, 2], 0) + \
242             self.create_player_list([400, 1000], [1, 2], [5, 7], [2, 3], 0) + \
243             self.create_player_list([400, 1000], [4, 6], [5, 7], [1, 1], 0) + \
244             self.create_player_list([400, 1000], [0, 1], [0.8, 2], [0, 2], 0) + \
245             self.create_player_list([400, 1000], [9, 10], [8, 12], [0, 2], 0) + \
246             self.create_player_list([400, 1000], [1, 2], [.7, 1.3], [0, 1], 0) + \
247
248             self.create_player_list([400, 1000], [3, 5], [.7, 1], [1, 4], 1) + \
249             self.create_player_list([400, 1000], [6, 7], [1, 1.1], [1, 2], 1) + \
250             self.create_player_list([400, 1000], [1, 2], [.8, 1], [4, 5], 1) + \
251             self.create_player_list([400, 1000], [0, 2], [.5, .7], [2, 3], 1) + \
252             self.create_player_list([400, 1000], [8, 9], [1, 1.2], [1, 1], 1) + \
253             self.create_player_list([400, 1000], [2, 4], [.6, 1.5], [2, 4], 1) + \
254
255             self.create_player_list([400, 1000], [4, 7], [0.3, 0.6], [1, 2], 2) + \
256             self.create_player_list([400, 1000], [5, 7], [.4, .5], [2, 5], 2) + \
257             self.create_player_list([400, 1000], [0, 3], [0.3, 0.4], [1, 3], 2) + \
258             self.create_player_list([400, 1000], [8, 10], [.7, .9], [4, 5], 2) + \
259             self.create_player_list([400, 1000], [9, 11], [1, 2], [0, 1], 2) + \
260             self.create_player_list([400, 1000], [2, 4], [.5, .7], [2, 3], 2) + \
261
262             self.create_player_list([400, 1000], [7, 9], [.4, .5], [3, 6], 3) + \
263             self.create_player_list([400, 1000], [2, 4], [.3, .4], [2, 4], 3) + \
264             self.create_player_list([400, 1000], [6, 10], [.5, .7], [5, 8], 3) + \
265             self.create_player_list([400, 1000], [0, 2], [0, .3], [6, 8], 3) + \
266             self.create_player_list([400, 1000], [3, 5], [.2, .3], [4, 6], 3) + \
267             self.create_player_list([400, 1000], [7, 8], [1, 1.3], [4, 5], 3) + \
268
269             self.create_player_list([400, 1000], [9, 12], [.1, .2], [2, 4], 4) + \
270             self.create_player_list([400, 1000], [5, 8], [.2, .3], [5, 7], 4) + \
271             self.create_player_list([400, 1000], [3, 5], [.1, .3], [4, 7], 4) + \
272             self.create_player_list([400, 1000], [2, 4], [.1, .2], [8, 12], 4) + \
273             self.create_player_list([400, 1000], [11, 16], [1, 4], [2, 7], 4) + \
274             self.create_player_list([400, 1000], [6, 10], [.3, .5], [7, 9], 4) + \
275
276             self.create_player_list([400, 1000], [6, 24], [0, 0], [5, 15], 5) + \
277             self.create_player_list([400, 1000], [6, 24], [0, .05], [5, 10], 5) + \
278             self.create_player_list([400, 1000], [16, 24], [0, .2], [4, 22], 5) + \
279             self.create_player_list([400, 1000], [12, 24], [0, .4], [5, 20], 5) + \

```

Page 4 of 6

Figure B.4

File - /Users/jonas/PycharmProjects/ml-identifying-problem-gamblers/decision_tree/tree.py

```

277         self.create_player_list([400, 1000], [16, 24], [1, 2], [2, 3], 5) + \
278         self.create_player_list([400, 1000], [0, 24], [0, .1], [20, 40], 5)
279
280     self.nrOfNodes = len(training_data)
281     print(self.nrOfNodes)
282     self.print_progress()
283
284     return self.build_tree(training_data)
285
286     def build_and_get_tree_small(self):
287         training_data = \
288             self.create_player_list([20, 70], [0, 3], [.8, 3], [0, 2], 0) + \
289             self.create_player_list([20, 70], [1, 2], [5, 7], [2, 3], 0) + \
290             self.create_player_list([20, 70], [4, 6], [5, 7], [1, 1], 0) + \
291             self.create_player_list([20, 70], [0, 1], [0.8, 2], [0, 2], 0) + \
292             self.create_player_list([20, 70], [9, 10], [8, 12], [0, 2], 0) + \
293             self.create_player_list([20, 70], [1, 2], [.7, 1.3], [0, 1], 0) + \
294 \
295             self.create_player_list([20, 70], [3, 5], [.7, 1], [1, 4], 1) + \
296             self.create_player_list([20, 70], [6, 7], [1, 1.1], [1, 2], 1) + \
297             self.create_player_list([20, 70], [1, 2], [.8, 1], [4, 5], 1) + \
298             self.create_player_list([20, 70], [0, 2], [.5, .7], [2, 3], 1) + \
299             self.create_player_list([20, 70], [8, 9], [1, 1.2], [1, 1], 1) + \
300             self.create_player_list([20, 70], [2, 4], [.6, 1.5], [2, 4], 1) + \
301 \
302             self.create_player_list([20, 70], [4, 7], [0.3, 0.6], [1, 2], 2) + \
303             self.create_player_list([20, 70], [5, 7], [.4, .5], [2, 5], 2) + \
304             self.create_player_list([20, 70], [0, 3], [0.3, 0.4], [1, 3], 2) + \
305             self.create_player_list([20, 70], [8, 10], [.7, .9], [4, 5], 2) + \
306             self.create_player_list([20, 70], [9, 11], [1, 2], [0, 1], 2) + \
307             self.create_player_list([20, 70], [2, 4], [.5, .7], [2, 3], 2) + \
308 \
309             self.create_player_list([20, 70], [7, 9], [.4, .5], [3, 6], 3) + \
310             self.create_player_list([20, 70], [2, 4], [.3, .4], [2, 4], 3) + \
311             self.create_player_list([20, 70], [6, 10], [.5, .7], [5, 8], 3) + \
312             self.create_player_list([20, 70], [0, 2], [0, .3], [6, 8], 3) + \
313             self.create_player_list([20, 70], [3, 5], [.2, .3], [4, 6], 3) + \
314             self.create_player_list([20, 70], [7, 8], [1, 1.3], [4, 5], 3) + \
315 \
316             self.create_player_list([20, 70], [9, 12], [.1, .2], [2, 4], 4) + \
317             self.create_player_list([20, 70], [5, 8], [.2, .3], [5, 7], 4) + \
318             self.create_player_list([20, 70], [3, 5], [.1, .3], [4, 7], 4) + \
319             self.create_player_list([20, 70], [2, 4], [.1, .2], [8, 12], 4) + \
320             self.create_player_list([20, 70], [11, 16], [1, 4], [2, 7], 4) + \
321             self.create_player_list([20, 70], [6, 10], [.3, .5], [7, 9], 4) + \
322 \
323             self.create_player_list([20, 70], [6, 24], [0, 0], [5, 15], 5) + \
324             self.create_player_list([20, 70], [6, 24], [0, .05], [5, 10], 5) + \
325             self.create_player_list([20, 70], [16, 24], [0, .2], [4, 22], 5) + \
326             self.create_player_list([20, 70], [12, 24], [0, .4], [5, 20], 5) + \
327             self.create_player_list([20, 70], [16, 24], [1, 2], [2, 3], 5) + \
328             self.create_player_list([20, 70], [0, 24], [0, .1], [20, 40], 5)
329
330     self.nrOfNodes = len(training_data)
331     print(self.nrOfNodes)
332     self.print_progress()
333
334     return self.build_tree(training_data)
335
336     def test(self):
337         testing_data = [
338             # time    win_loss    deposits    answer
339             [1,      1,          1,          0],
340             [6,      5,          1,          0],
341             [9,      9,          2,          0],
342             [2,      .9,         1,          0],
343             [4,      1,          4,          1],
344             [1,      .6,         2,          1],
345             [8,      1.1,        1,          1],

```

Page 5 of 6

Figure B.5

```

File - /Users/jonas/PycharmProjects/ml-identifying-problem-gamblers/decision_tree/tree.py
346         [7, 1.05, 2, 1],
347         [4, .5, 2, 2],
348         [3, .6, 3, 2],
349         [6, .45, 4, 2],
350         [10, 2, 0, 2],
351         [4, .4, 6, 3],
352         [9, .55, 7, 3],
353         [7, 1.3, 5, 3],
354         [8, .61, 5, 3],
355         [12, 2, 10, 4],
356         [3, .3, 7, 4],
357         [7, .32, 7, 4],
358         [5, .4, 9, 4],
359         [12, .3, 20, 5],
360         [1, 0, 30, 5],
361         [17, 1.3, 2, 5],
362         [0.5, .05, 34, 5],
363     ]
364     for row in testing_data:
365         print("Actual: %s. Predicted: %s" %
366               (row[-1], self.print_leaf(self.classify(row, self.tree))))
367
368     def get_actual(self, data):
369         return list(self.classify(data, self.tree).keys())[0]
370
371     def print_progress(self):
372         progress1 = int(((self.builtNodes / self.nrofNodes) * 100) / 5)
373         progress2 = ((self.builtNodes / self.nrofNodes) * 100)
374         sys.stdout.write('\rBuilding Tree: [{0}{1}] {2}%'.format(
375             '█' * progress1, '-' * int((100 / 5) - progress1), round(progress2, 1)
376         ))
377         if progress2 == 100:
378             print()
379

```

Page 6 of 6

Figure B.6

C Decision Node

File - /Users/jonas/PycharmProjects/ml-identifying-problem-gamblers/decision_tree/decisionnode.py

```
1 class DecisionNode:
2     """A Decision Node asks a question.
3
4     This holds a reference to the question, and to the two child nodes.
5     """
6
7     def __init__(self,
8                 question,
9                 true_branch,
10                false_branch):
11         self.question = question
12         self.true_branch = true_branch
13         self.false_branch = false_branch
14
```

Page 1 of 1

Figure C.1

D Question

File - /Users/jonas/PycharmProjects/ml-identifying-problem-gamblers/decision_tree/question.py

```
1 from utils import is_numeric
2
3
4 class Question:
5     """A Question is used to partition a dataset.
6
7     This class just records a 'column number' (e.g., 0 for Color) and a
8     'column value' (e.g., Green). The 'match' method is used to compare
9     the feature value in an example to the feature value stored in the
10    question. See the demo below.
11    """
12
13    def __init__(self, column, value, header):
14        self.column = column
15        self.value = value
16        self.header = header
17
18    def match(self, example):
19        val = example[self.column]
20        if is_numeric(val):
21            return val >= self.value
22        else:
23            return val == self.value
24
25    def __repr__(self):
26        condition = "="
27        if is_numeric(self.value):
28            condition = ">="
29        return "Is %s %s %s?" % (
30            self.header[self.column], condition, str(self.value))
31
```

Page 1 of 1

Figure D.1

E Leaf

File - /Users/jonas/PycharmProjects/ml-identifying-problem-gamblers/decision_tree/leaf.py

```
1 from decision_tree import tree
2
3
4 class Leaf:
5     """A Leaf node classifies data.
6
7     This holds a dictionary of class (e.g., "Apple") -> number of times
8     it appears in the rows from the training data that reach this leaf.
9     """
10
11     def __init__(self, rows):
12         self.predictions = tree.class_counts(rows)
```

Page 1 of 1

Figure E.1

F Player/Gambler

File - /Users/jonas/PycharmProjects/ml-identifying-problem-gamblers/decision_tree/player.py

```
1 class Player:
2
3     def __init__(self, time, win_rate, nr_deposit, answer):
4         self.time = time
5         self.win_rate = win_rate
6         self.nr_deposit = nr_deposit
7         self.answer = answer
8
9     def get_player_params(self):
10         return [self.time, self.win_rate, self.nr_deposit, self.answer]
11
12
```

Page 1 of 1

Figure F.1

G Mail

File - /Users/jonas/PycharmProjects/ml-identifying-problem-gamblers/mail/mail.py

```
1 import smtplib
2 import ssl
3 from email.mime.image import MIMEImage
4 from email.mime.multipart import MIMEMultipart
5 from email.mime.text import MIMEText
6
7
8 class Mail:
9
10     def __init__(self):
11         self.context = ssl.create_default_context()
12         self.port = 465
13         self.sender_email = "info@sourceempire.io"
14         self.SMTP = "mail.privateemail.com"
15         print('Login\nEmail: {}'.format(self.sender_email))
16         self.password = input("Password: ")
17         print()
18
19     def send_mail(self, receiver_email, result):
20         message = MIMEMultipart('related')
21         message["Subject"] = "Your result - PathoGamble"
22         message["From"] = "info@sourceempire.io"
23         message["To"] = receiver_email
24         message.preamble = 'This is a multi-part message in MIME format.'
25
26         message_alternative = MIMEMultipart('alternative')
27         message.attach(message_alternative)
28
29         message_text = MIMEText('This is the alternative plain text message.')
30         message_alternative.attach(message_text)
31
32         message_text = MIMEText(''
33                                 <!DOCTYPE html>
34                                 <html>
35                                     <head>
36                                         <style type="text/css">
37                                             .mail-container {{
38                                                 width: 100%;
39                                                 text-align: center;
40                                                 border: 1px solid lightblue;
41                                                 box-sizing: border-box;
42                                             }}
43                                             .mail-container .image {{
44                                                 height: 200px;
45                                                 width: 200px;
46                                                 margin: 20px auto;
47                                             }}
48                                             .mail-container .image img {{
49                                                 width: 200px;
50                                                 height: 200px;
51                                             }}
52                                             .mail-container .result-header {{
53                                                 font-family: 'Georgia', serif;
54                                                 font-size: 25px;
55                                                 font-weight: bold;
56                                             }}
57                                             .mail-container .result-text {{
58                                                 font-family: sans-serif;
59                                                 font-size: 13px;
60                                             }}
61                                         </style>
62                                     </head>
63                                     <body>
64                                         <div class="mail-container">
65                                             <div class="image">
66                                                 
67                                             </div>
68                                             <p class="result-text">{}</p>
69                                         </div>
```

Page 1 of 3

Figure G.1

File - /Users/jonas/PycharmProjects/ml-identifying-problem-gamblers/mail/mail.py

```

70         </body>
71     </html>
72     '''format(switch(result)), 'html')
73     message_alternative.attach(message_text)
74
75     fp = open('./resources/images/logo.png', 'rb')
76     message_image = MIMEImage(fp.read())
77     message_image.add_header('Content-ID', '<image1>')
78     message.attach(message_image)
79     fp.close()
80
81     with smtplib.SMTP_SSL(self.SMTP, self.port, context=self.context) as server:
82         server.login(self.sender_email, self.password)
83         server.sendmail(self.sender_email, receiver_email, message.as_string())
84
85
86 def switch(result):
87     switcher = {
88         0: '''
89             <p class="result-header">
90                 Your result: 0/5
91             </p>
92             <p class="result-text">
93                 We see no reason for you to get help. Your behaviour
94                 does not seem to be destructive.
95                 Nevertheless, if you feel like you need help, you can call 07X-XXX XX XX
96                 and tell them about your problems. They will be happy to help.
97             </p>
98         ''',
99         1: '''
100             <p class="result-header">
101                 Your result: 1/5
102             </p>
103             <p class="result-text">
104                 You have been given a low chance of developing a gambling addiction.
105                 Although, we can see a pattern in your behaviour, just think
106                 about how much you play and you will be fine.
107                 If you feel like you have a problem, call 07X-XXX XX XX, they will be happy to guide
108                 you.
109             </p>
110         ''',
111         2: '''
112             <p class="result-header">
113                 Your result: 2/5
114             </p>
115             <p class="result-text">
116                 You have been given a low to mediate risk of developing a gambling addiction.
117                 Beware, try to gamble a bit less or spend a bit less money. Feel free to call
118                 07X-XXX XX XX if you have questions about problem gambling, they are happy to help!
119             </p>
120         ''',
121         3: '''
122             <p class="result-header">
123                 Your result: 3/5
124             </p>
125             <p class="result-text">
126                 You have been given a mediate to high risk
127                 risk of developing a gambling addiction.
128                 Does surrounding people think that you gamble to much?
129                 Try to gamble a bit less and call 07X-XXX XX XX if you
130                 need someone to talk to.
131             </p>
132         ''',
133         4: '''
134             <p class="result-header">
135                 Your result: 4/5
136             </p>
137             <p class="result-text">

```

Page 2 of 3

Figure G.2

File - /Users/jonas/PycharmProjects/ml-identifying-problem-gamblers/mail/mail.py

```
138         You have been placed very high on the
139         risk-scale of developing a gambling addiction.
140         We suggest that you call 07X-XXX XX XX for suggestions
141         on how to gamble less.
142     </p>
143     ...,
144     5: ...,
145         <p class="result-header">
146             Your result: 5/5
147         </p>
148         <p class="result-text">
149             You got the absolute highest level of
150             risk to develop a gambling addiction.
151             We suggest that you stop gambling immediately and
152             try and get help. Our tip is that you call 07X-XXX XX XX
153         </p>
154     ...,
155 }
156 return switcher.get(result)
157
```

Page 3 of 3

Figure G.3