**CMP\_SC 4770: Intro to Computational Intelligence**

**Fall 2022 – Project 2: Fuzzy Sets**

**By: Mikey Joyce**

**PAWPRINT: MPJYKY**

**11/1/2022**

**Table of Contents:**

1. Introduction Page 1
2. Project Description Page 1-5
3. Implementation Page 5-6
4. Experiments and Results Page 7-22
5. Conclusion Page 22

**Introduction:**

In project 2 we will discuss and implement a Mamdani style fuzzy logic system. The system will be utilized to classify long or short positions for stock market data. It will analyze the Bid-Ask Spread along with the Beta, which is a numeric volatility identifier, to produce an indication if we should be trading the stock based on these metrics. The 50 day and 200 day moving average along with the relative strength index will also be analyzed to determine whether or not we should be entering a long or short position. Finally, the implementation will then aggregate these two metrics to get the final long or short indication. We will also utilize the fuzzy library in a non-classification manner to identify a fair market value for a Gibson Les Paul. We will use the metrics of age, model type, and last price and aggregate these to produce what the fair market value of the guitar is.

**Project Description:**

In this section we will discuss the theoretical building blocks we need for our Mamdani fuzzy logic system. We will talk about what it means to be a fuzzy value. Then we will talk about the methods, or building blocks, that we need for our fuzzy system. Lastly, we will define what a fuzzy logic system is.

**What does it mean to be fuzzy?**

First, we will discuss what it means to be a fuzzy value. A fuzzy value is some value on the real number set that is between 0 and 1. These values represent varying degrees of membership or truth. For instance, say we are trying to determine if the weather temperatures are hot. If it is 100 degrees outside, we can assign this a fuzzy value of 1 because it is with absolute certainty that it is hot outside. Say it is 20 degrees outside, we can assign this a fuzzy value of 0 because it is with absolute certainty that it is not hot outside. However, say we have a temperature of 75 degrees outside. There is some uncertainty on if it is hot or not outside. Because of this we would have to assign it a fuzzy value that shows that there is some degree of truth for classifying 75 degrees as being hot. For this example, we can give it an arbitrary fuzzy value of 0.6, which would represent it being 60% hot outside. Now that we understand what fuzzy values are, we must move on to what methods are required to create a fuzzy system.

**Fuzzy Building Blocks**

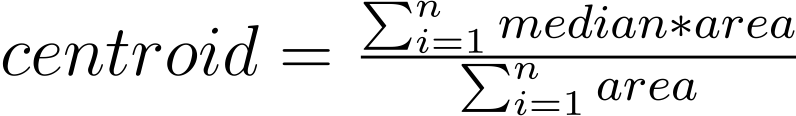
Here we will discuss the various building blocks we need for our fuzzy system. Later in the project we will program these building blocks into a library that we can utilize in the implementation phase.

So, what does our fuzzy system need to work? First, we need membership functions. There are various different types of membership functions, such as the triangle membership function, the trapezoid membership function, and the gaussian membership function. But why do we need membership functions? We need membership functions simply because we need to be able to represent various degrees of truth for a classifier based on the given domain. So, again we will discuss the temperature problem. Our domain let’s say is 0 degrees to 100 degrees (I am aware that the temperature can go below 0 degrees and above 100 degrees, but since its rare here in Missouri we can omit those from this example). We need to be able to represent what temperatures are cold, medium, and hot. That’s where the membership functions come in. Using a trapezoid membership function we can determine that 0-32 degrees have a fuzzy value of 1 for cold meaning they are absolutely cold. Then, say we can start tapering off cold from 32-50. So, during this portion of the membership there are varying degrees of truth for if it is cold or not. If we pick 41 degrees it would have a fuzzy value of 0.5, which means that it is 50% cold at 41 degrees. The membership functions essentially allow us to classify data on our domain.

The next building block that is required for our system is the rule set. One might ask, why do we need a rule set? Well, if we have a rule set then we are able to infer outputs based on our inputs. We want to be able to do this that way we can combine multiple different membership functions and apply a result on a completely different membership function. For instance, let’s use the temperature problem combined with a weather problem to determine whether or not we should go outside. Say the hot membership function is firing at 1 for the temperature domain. Let’s also say that the sunny membership function is firing at 1 for the weather domain. We need a rule here that says, IF it is hot AND sunny THEN output on the decision domain: go outside. Without rules we can’t combine different membership functions on separate domains to give us a meaningful result.

After this, we need to aggregate these results obtained from the rule set. Why should we aggregate the results? Well, if we take the max of the result of all of the rules, we can then identify which membership function our result most identifies with. This is important because we are trying to obtain a meaningful answer at the end of the day.

Lastly, our final building block will be the defuzzification mechanism. We need a defuzzification mechanism because we initially had a crisp input, which is any number on the real number set, and we want our output to also be crisp and not fuzzy. Using the centroid method, we take in our bounds for the domain and divide it up into subareas based on the length of the bounds. We then find the median and area of each subarea. After that we take the summation of the current median multiplied by the current area and divide this result by the summation of the area to get our crisp result. The equation for the centroid method for a discrete membership function is below:

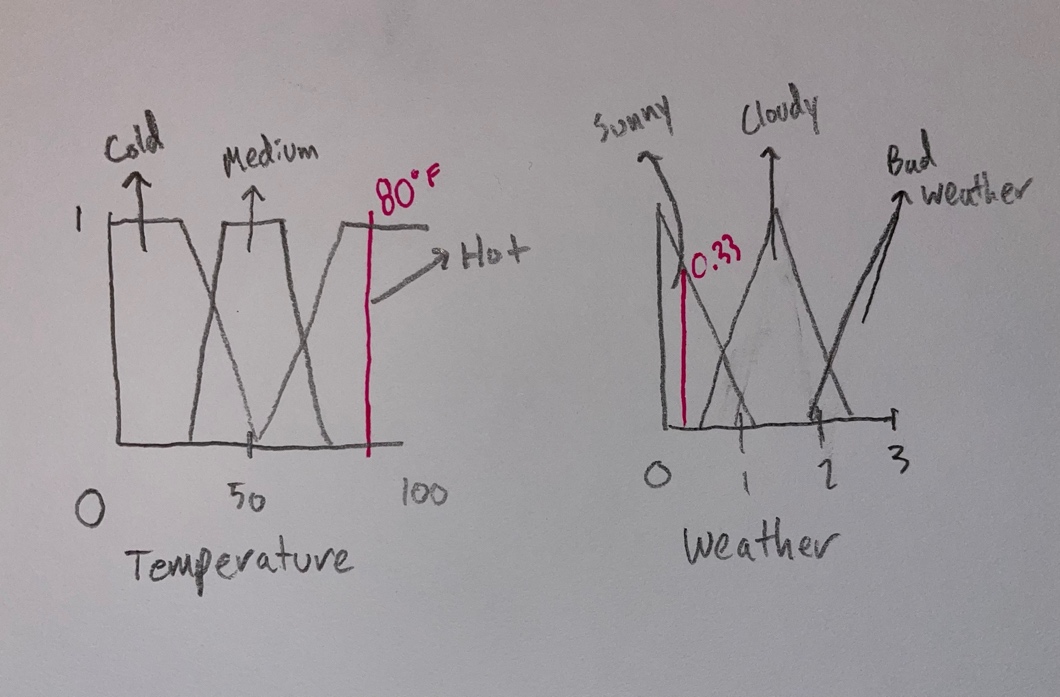


***Figure 2.1*** *shows the centroid equation*

Take note that the centroid equation, when used with a continuous membership function, will look a little different. Instead of taking the summation of these you will take the integral of them. In the code it still remains the same. The last topic in this section that needs to be discussed is what is a fuzzy logic system?

**Fuzzy Logic System**

What is a fuzzy logic system? Basically, it is a system that uses the above building blocks to try and take a crisp input and use reasoning, similar to something a human would do with the temperature/weather problem, via a fuzzy number to obtain a crisp result. How would this look? Well let’s do an example using the temperature/weather problem. The following shows the input domains with their membership functions and our target input data:



***Figure 2.2*** *shows a temperature and weather input example*

Now we must take these inputs and push them through our inference engine, which is just a fancy term for our rule base. We will only discuss the rule that should target our current input. That rule is IF it is hot AND sunny THEN go outside. Then, we need to aggregate the results from the rules and pick the rule that was targeted the most. After this rule is targeted, we need to defuzzify our result to get a crisp value. The following shows this happening on a graph:

Diagram

Description automatically generated

***Figure 2.3*** *shows a temperature and weather output example*

All in all, in this section we discussed the topic that the project is based on. We did this by exploring what it means to be fuzzy, identifying the building blocks necessary for our project, and determining what a fuzzy logic system is.

**Implementation:**

The charts in this section represent the flow of the code. If the node on the chart is red, that means it references our fuzzy library.

**Diagram

Description automatically generated**

***Figure 3.1*** *shows the flowchart for the stock trading indicator*

Diagram

Description automatically generated

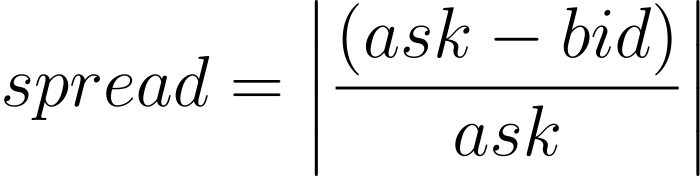
***Figure 3.2*** *shows the flowchart for the Gibson Les Paul fair value analyzer*

**Experiments and Results:**

In this section we will discuss the experiments and results that we conducted with the implementations of our fuzzy logic system using the stock market indicator classifier and the non-classification guitar fair market value analyzer.

First, let’s discuss the classification task that was completed with creating a stock market indicator using the fuzzy library. This indicator uses 4 metrics to decide whether or not to open a long position, do nothing, or open a short position.

The first two metrics that are contained in this indicator are the beta value and the bid-ask spread. The beta value is a numeric representation of the volatility of a stock. A value of 1 would mean the stock has an equal volatility to the S&P. A value of less than 1 would mean the stock is less volatile than the S&P. A value of greater than 1 would mean the stock is more volatile than the S&P. Currently, when writing this report, the highest beta stock is SM Energy at 4.65. Because there is no real max bound for this indicator, we will say for simplicities sake that if a stock has a beta of greater than 2 then we will just say their beta is 2. We can do this because we will be using the gaussian membership function, yet we still want any beta that is 2 or greater to be at the peak of the gaussian curve and instead of combing a gaussian and trapezoid membership function into one, it made more sense to just set the max bound for beta. The next indicator that will be used in conjunction with the beta volatility index is the bid-ask spread. The following equation shows how the bid ask spread is calculated:



***Figure 4.1*** *shows the equation for the bid ask spread*

Since the bid and ask price will ultimately be quite close to each other, we know that the upper bound of this indicator cannot surpass the value of 1. Since we are taking the magnitude, the lower bound will be 0. Values for the bid-ask spread that are less than 0.015 are seen as assets with high demand compared to their supply, which we will describe as a “good” spread. Assets that are higher than 0.05 are ones that we really don’t want to mess around with, so we will describe these assets as a “bad” spread. It makes sense to use the trapezoid membership function for this indicator because we want a peak for the indicator that lasts for more than one value on the x-axis because every asset with a value that is greater than 0.05 is 100% bad.

Lastly, we need to define a universe to aggregate these indicators to. I named this universe the beta spread fusion universe. Its bounds are from 0-100 to represent a percentage of how fit the stock is for trading, the closer to 100 the better it is. The triangle membership function was used for this universe because we wanted only one value to be the peak of each membership and we wanted a steady increase from the beginning value to the end value because the 70% vs 71% is not exponentially better than each other when considering a confidence level.

A picture containing line chart

Description automatically generated

***Figure 4.2*** *shows the universes and membership functions for beta, spread, and the fusion*

Now, we need to discuss the rules that we want to actually perform the inference. Since there are four different combinations for the input it makes sense to have four rules to cover it. The rules are:

1. IF there is high volatility AND the spread is bad THEN return average
2. IF there is high volatility AND the spread is good THEN return good
3. IF there is low volatility AND the spread is bad THEN return bad
4. IF there is low volatility AND the spread is good THEN return average

Lastly, we need to do some tests to make sure the rules are firing when we want them to. The first test I am going to do will be demonstrated throughout this section. This test was taking data for Tesla’s stock price on October 17th, 2022. The other tests that I will do will be arbitrary numbers that is testing whether or not our variables are firing or not, for these tests I will not show the graphs, only the graph will be shown for the Tesla test.

For the Tesla test we have Beta set at 2.13 and marked down to 2 for simplicity. We also have a bid price of $222.75 and an ask price at $221.95, so the spread will be set at 0.0036. Since the beta has max membership value for high and the spread also has max membership, we are expecting something that has a high confidence level for trading this stock. AKA, we want to display that this stock is fit for trading. Here are the results:

Chart

Description automatically generated

***Figure 4.2*** *results for the aggregation of beta and spread*

The percent confidence value that was returned from the aggregation was: 83.333% which identifies with the best membership function. We tested rule 2 above now we need to test the other three rules. The following is a rundown of three more tests that show that our rules are firing correctly:

* Rule 1 test:
  + Beta = 1.85, Spread = 0.2
  + Expectation: return a percentage that identifies with the average membership function
  + Result:
    - Percentage = 49.999% which identifies with the average membership function
    - Pass!
* Rule 3 test:
  + Beta = 0.4, Spread = 0.5
  + Expectation: return a percentage that identifies with the bad membership function
  + Result:
    - Percentage = 18.563% which identifies with the bad membership function
    - Pass!
* Rule 4 test:
  + Beta = 0.2, Spread = 0.001
  + Expectation: return a percentage that identifies with the average membership function
  + Result:
    - Percentage = 50.000% which identifies with the average membership function
    - Pass!

Now, we must discuss the other 2 metrics used in this fuzzy logic system. These metrics are traditional stock indicators. The first one is the moving average, which is calculated by taking the 50-day moving average and the 200-day moving average and doing the following calculation:

***Crossover = (50-day ma) – (200-day ma)***

The 50-day moving average is the average price for the stock for the past 50 days, (the same calculation is made for the 200 day except instead of 50 days its 200 days). The other indicator we will be using is the relative strength index (RSI). The relative strength index is a momentum indicator with bounds 0-100. Values between 50-75 means its trending upwards. Values above 75 means it’s likely overbought, aka too much pressure following the momentum and could be likely for reversal. Values between 50-25 means its trending downwards. Lastly, values below 25 mean it’s likely oversold. Since the peaks of trending up and trending down are in the middle, we want a membership function that has one peak. So, for the RSI we will pick a triangle membership function. Here is what the membership functions and universes look like:

Line chart

Description automatically generated

***Figure 4.3*** *shows the universes and membership functions for crossover, rsi, and the fusion*

Now, we need to discuss the rules that we want to actually perform the inference. Since there are nine different combinations for the input it makes sense to have nine rules to cover it. Except it really doesn’t. If there are values near the middle of the crossover or the RSI, we want to immediately put those rules into the average section of the fusion. That leaves us with six rules. The rules are:

1. IF crossover is buy AND RSI is up THEN return good
2. IF crossover is sell AND RSI is up THEN return average
3. IF crossover is buy AND RSI is down THEN return average
4. IF crossover is sell AND RSI is down THEN return bad
5. IF crossover is neutral THEN return average
6. IF RSI is neutral THEN return average

Let’s conduct some tests to make sure the rules are firing as expected. The tests will be conducted in the same manner as the last ones. For the Tesla data we have: 50ma = $278.18, 200ma = $285.47, moving average = $-7.29, and RSI = 36.4. Since the moving average is within the bounds of the neutral membership function the expected result is to return an average value in the crossover RSI fusion universe which would test rule #5. Here are the results:

Shape

Description automatically generated

***Figure 4.4*** *results for the aggregation of beta and spread*

The resulting value was 46.958% which is within the bounds of the average membership function on the aggregated universe. Let’s test the remaining 5 rules now:

* Rule 1 test:
  + Crossover = 99, RSI = 62
  + Expectation: return a percentage that identifies with the good membership function
  + Result:
    - Percentage = 80.368% which identifies with the good membership function
    - Pass!
* Rule 2 test:
  + Crossover = -74.20, RSI = 75
  + Expectation: return a percentage that identifies with the average membership function
  + Result:
    - Percentage = 50.002% which identifies with the average membership function
    - Pass!
* Rule 3 test:
  + Crossover = 45.23, RSI = 5
  + Expectation: return a percentage that identifies with the average membership function
  + Result:
    - Percentage = 49.897% which identifies with the average membership function
    - Pass!
* Rule 4 test:
  + Crossover = -54.99, RSI = 28
  + Expectation: return a percentage that identifies with the bad membership function
  + Result:
    - Percentage = 19.200% which identifies with the bad membership function
    - Pass!
* Rule 6 test:
  + Crossover = 75, RSI = 45
  + Expectation: return a percentage that identifies with the average membership function
  + Result:
    - Percentage = 49.998% which identifies with the average membership function
    - Pass!

So, now we are left with two crisp values. We need to obtain a universe to aggregate these crisp values to get our result. The decision universe will have the same bounds 0-100 as the other two and use the triangle membership functions just the way that they do. It’s worth it to note that the no-trade membership function on the decision is somewhat arbitrary in its bounds. If this system were to ever be used in the industry, the no-trade membership bounds may have to be tweaked. For instance, with the current bounds an order could be submitted for Long or Short with only 30% confidence for that trade’s success. To fix an issue like this we could widen the bounds for the no-trade membership function, and if we really only want it to submit orders with high confidence, we may even have to change the membership function to something like a trapezoid membership function. The following graph is the final universe:

Line chart

Description automatically generated

***Figure 4.5*** *shows the universes and membership functions for beta/spread fusion, crossover/RSI fusion, and the decision*

Now let’s talk rules for this final aggregation. There are nine different ways to combine the two and we can cut two rules out because if the beta/spread fusion is bad for any reason then no trade will be made. Here is the rule set:

1. IF beta/spread is neutral AND crossover/RSI is good THEN return Long
2. IF beta/spread is neutral AND crossover/RSI is average THEN return No-trade
3. IF beta/spread is neutral AND crossover/RSI is bad THEN return Short
4. IF beta/spread is best AND crossover/RSI is good THEN return Long
5. IF beta/spread is best AND crossover/RSI is average THEN return No-trade
6. IF beta/spread is best AND crossover/RSI is bad THEN return Short
7. IF beta/spread is worst THEN return No-trade

So, we must conduct 7 tests to make sure our fuzzy logic system is working correctly. We will start with the Tesla example first of course. With the Tesla example we got beta/spread = 83.333 and crossover/RSI = 46.958. Our expectation is that rule #5 will fire and we will get a result in the No-trade class. Meaning if an industry bot were to use this indicator, it would not make a trade at this moment in time. Here are the results:

Chart, diagram

Description automatically generated

***Figure 4.6*** *results for the aggregation of beta/spread and crossover/RSI*

The result that the fuzzy logic system gave us was No Trade. This is good because it was the expected result. If we look at Tesla’s market action since October 17th it appears to have flatlined and made no real large moves, so under this circumstance the indicator actually yielded positive results. Obviously, more testing during a live scenario would have to be done to determine the accuracy of the system.

A picture containing chart

Description automatically generated

***Figure 4.7*** *shows Tesla’s market action. October 17th price = 219.35. Price on October 26th = 224.64. Note in the code the bid/ask doesn’t exactly correlate with the close price on the 17th because the data was taken in the middle of the day.*

Now, we need to test the remaining six rules of the fuzzy logic system to make sure it is working the way we are expecting it to. Here are the results:

* Rule 1 test:
  + Beta/Spread = 50, Crossover/RSI = 84.564
  + Expectation: return a classification of long
  + Result:
    - Long
    - Pass!
* Rule 2 test:
  + Beta/Spread = 45, Crossover/RSI = 62
  + Expectation: return a classification of no-trade
  + Result:
    - No-trade
    - Pass!
* Rule 3 test:
  + Beta/Spread = 54, Crossover/RSI = 12
  + Expectation: return a classification of short
  + Result:
    - Short
    - Pass!
* Rule 4 test:
  + Beta/Spread = 99, Crossover/RSI = 78
  + Expectation: return a classification of long
  + Result:
    - Long
    - Pass!
* Rule 6 test:
  + Beta/Spread = 81, Crossover/RSI = 34
  + Expectation: return a classification of short
  + Result:
    - Short
    - Pass!
* Rule 7 test:
  + Beta/Spread = 1, Crossover/RSI = 100
  + Expectation: return a classification of no-trade
  + Result:
    - No-trade
    - Pass!

Here we have shown that for every possible scenario our fuzzy logic system classifies the signals the way that we are expecting them to. Last, but not least, we will talk about the non-classification implementation task for our fuzzy logic system which is the Gibson Les Paul fair market analyzer.

The Gibson Les Paul guitar is a guitar that has a wide range of values based on different metrics. The [cheapest brand new](https://www.sweetwater.com/c590--Solid_Body?highlight=LPSPTP01WWCH&mrkgadid=3344197479&mrkgcl=28&mrkgen=gpla&mrkgbflag=0&mrkgcat=bass&acctid=21700000001645388&dskeywordid=92700058100046332&lid=92700058100046332&ds_s_kwgid=58700006436060614&ds_s_inventory_feed_id=97700000007215323&dsproductgroupid=780235072990&product_id=LPSPTP01WWCH&prodctry=US&prodlang=en&channel=online&storeid=&device=c&network=g&matchtype=&adpos=largenumber&locationid=1020268&creative=473938416872&targetid=aud-371863008218:pla-780235072990&campaignid=11404733931&awsearchcpc=1&gclid=CjwKCAjw5P2aBhAlEiwAAdY7dPp3CL9LJYoChgFSmCu0TRQWYn-R93lnGFEVXrfwt7UQ3mnopPb5oxoCjfUQAvD_BwE&gclsrc=aw.ds) Les Paul made by Gibson that you can buy today is a $1000 guitar that is pretty bare bones, but solid, nonetheless. But there have been Les Paul’s that have sold for more than $300,000. With these wide range of values, it is hard for someone who doesn’t know about guitars to price their instrument to sell at a fair value. For instance, say a loved one just passed and you, a non-guitar expert, just inherited an authentic 1959 Les Paul Standard. Without much knowledge you may accidentally give someone a steal on this guitar. You may think $1000-$3000 seems fair because your loved one really loved this guitar, and you really just want to get rid of it because you don’t play guitar. Or maybe you just throw it in storage and don’t pay much attention to it. Well, it turns out there is a possibility this guitar could be priced at a value comparable to a house. Here is an example: [1959 Les Paul](https://reverb.com/item/59847657-gibson-les-paul-standard-1959-burst).

The goal of this fuzzy system is to take in 3 different metrics and aggregate them to give us a price that should roughly be the fair market value for said instrument. The metrics that we will discuss will be the age, model, and the last price. The age is by far the biggest factor in price. Some may ask, why is age the most important factor? Well, there is a couple of reasons. Reason number one is that since its old other guitars its age have fallen apart and been broken. This makes the guitar rarer because the supply is so low for authentic old guitars. Another reason the age is important is because the older instruments are the instruments that guitar legends such as Hendrix or Jimmy Page played. Lots of guitarists like to imitate their idols so that they can try their best to sound like them. The older the guitar the more expensive the guitar will be, it’s almost an exponential metric. The next metric we will discuss will be the model. We will include three different models in this example which are, the standard model, the pro model, and the custom model. The standard model is the bare bones model and is usually the cheaper portion of the models when sold new. The pro model is similar to the standard with a couple of extra features that are deemed as nice, because of this is commands a slightly more expensive price, especially in the younger age group of guitars. Lastly, we have the custom model. This model is much different from the other two because custom models are unique guitars. Having a custom model is the second most important factor in having a higher priced guitar. These guitars will sell at prices such as $10,000 new and could appreciate in value. The last metric we will provide in this value analyzer will be the last price. This is a crucial metric because it will determine how much somebody is willing to pay for a guitar. If the guitar that the seller is selling was bought as used then we can say that the guitar is most likely going to hold this value at least, if not appreciate a bit in value. But, if the seller is selling a guitar that they bought new, its likely possible the guitar decreased in value unless it was bought as a custom model or bought a couple decades ago. Take note that we are only going over the most important metrics that determine the fair value price of a Les Paul. There are things such as the paint finish, more specific types of models, etc. that could affect the price. Now that we understand the basic rundown of how age, model, and last price could affect the fair value of a guitar, it’s time to discuss the details of the experimentation of this fuzzy system.

Graphical user interface

Description automatically generated with low confidence

***Figure 4.8*** *shows the universes and membership functions for age, model, last price and the fair market value*

The first membership function is the age. We used the trapezoid membership function because we wanted multiple ages to have the maximum fuzzy value that corresponds with their age. The bound for the age membership function is 70 years old because the first Gibson Les Paul model was made in 1952 which was 70 years ago. The next membership function is model. We used triangle membership function because we want there to only be one value for maximum fuzzy value. We did this because the value 0 will correspond with the standard model, 45 will be pro, and 70 will be custom. Initially, I wanted the values to be 0, 1, and 2 because that makes the most sense, but matplotlib gave me errors if my first two membership universes didn’t have the same exact bounds. That detail doesn’t really matter that much because if this were ever implemented for the public most likely the input for these models will be a dropdown menu that way we could assign it the values of 0, 45, and 70 automatically in the code. The next membership function will be the last price. This uses the trapezoid membership function again because we want multiple prices to be maximum cheap, average, and expensive. The bounds are $0-$300,000. For cheap, the bounds will be $0-$4000, with $0-$3000 representing the maximum cheapness. For average, the bounds will be $3500-$9000, with $4000-$8000 representing the maximum averageness. Lastly for expensive, the bounds will be $7500-$300,000, with $10,000-$300,000 being maximum expensiveness. The last membership universe is for the aggregated value. This universe is similar to the last price universe because it has the same bounds and cheap and average membership bounds are also the same. The only differences are that expensive fair value has bounds $7500-$100,000, with $10,000-$50,000 representing maximum expensiveness. The last difference is that we have another membership function named rare. It made sense to add the rare membership function because not many guitars will fall within this price range unless they are older guitars, which are rare guitars. The bounds for rare guitars are $50,000-$300,000, with $80,000-$300,000 representing maximum rareness.

Now, we can discuss the rules that perform our inference. There are a lot of rules for each age group. Initially, I tried to only have a few rules per age group, but it would result in the edge cases getting wrongly assigned. Old guitars used to have 2 rules, but now have 4. Middle aged guitars used to have 4 rules, but now have 6. Lastly, young guitars used to have 8 rules, but now have 6. Here are the rules:

For old guitars:

1. IF model is standard AND last price is expensive OR average THEN return rare
2. IF model is pro AND last price is expensive OR average THEN return rare
3. IF model is custom AND last price is expensive OR average THEN return rare
4. IF last price is cheap THEN return expensive

For middle aged guitars:

1. IF model is standard AND last price is expensive THEN return expensive
2. IF model is pro AND last price is expensive THEN return expensive
3. IF model is custom AND last price is expensive OR average THEN return expensive
4. IF model is standard AND last price is average OR cheap THEN return average
5. IF model is pro AND last price is average OR cheap THEN return average
6. IF model is custom AND last price is cheap THEN return average

For young guitars:

1. IF model is standard AND last price is expensive THEN return average
2. IF model is pro AND last price is expensive THEN return average
3. IF model is custom AND last price is expensive THEN return expensive
4. IF model is standard AND last price is average OR cheap THEN return cheap
5. IF model is pro AND last price is average OR cheap THEN return cheap
6. IF model is custom AND last price is average OR cheap THEN return average

Take note that the statement “for old/middle aged/young guitars” is the first antecedent for each rule. It seemed like it would be redundant and hard to read if I typed out “IF age is old AND” for each rule.

Now we will demonstrate that our fuzzy logic system is doing what is expected by running some experiments. We will run 15 experiments to make sure each rule is doing what is expected. Like the stock indicator, for the first experiment we will show a graphical result via a graph. The remaining results will be shown through data. The first one we will try and target an older guitar that was last sold at an unfair price: age = 59 years old (1963), model = standard, and last price = $3000. We are expecting it to return a fair price within the expensive membership bounds which would test rule #4. Here is the result:

Chart

Description automatically generated

***Figure 4.9*** *results for the aggregation result of age, model, and last price to give us a fair market value. The price returned = $43,443.40*

The fair market price that was given for this specific guitar was $43,443.40. Look at that return on investment! It gave us the correct membership identification based on what we were expecting, which was an expensive price assignment. Now, let’s take a look at the other rules and results when testing them:

* Rule 1 test:
  + Age = 53, model = standard, last price = 7999.99 (average/expensive)
  + Expectation: return a price within the bounds of rare
  + Result:
    - Fair value price: $ 156,591.03 which is rare
    - Pass!
* Rule 2 test:
  + Age = 70, model = pro, last price = $8050 (average/expensive)
  + Expectation: return a price within the bounds of rare
  + Result:
    - Fair value price: $ 181,980.94 which is rare
    - Pass!
* Rule 3 test:
  + Age = 62, model = custom, last price = $7000 (average)
  + Expectation: return a price within the bounds of rare
  + Result:
    - Fair value price: $ 182,339.93 which is rare
    - Pass!
* Rule 5 test:
  + Age = 30, model = standard, last price = $10,000 (average)
  + Expectation: return a price within the bounds of expensive
  + Result:
    - Fair value price: $ 43,443.40 which is expensive
    - Pass!
* Rule 6 test:
  + Age = 38, model = pro, last price = $11,053.23 (average)
  + Expectation: return a price within the bounds of expensive
  + Result:
    - Fair value price: $ 43,443.40 which is expensive
    - Note: to me it’s a bit strange that this gave us the same result as rule 5 despite using different numbers. My prediction is that it’s because the expensive membership bounds are massive compared to cheap and average. Let’s do another test with a more expensive price, but same age and model.
    - Last price = $100,000
      * Fair value price: $ 43,443.40
      * Still hasn’t changed which is interesting. It could be possible that $43,443.40 is the maximum bound for guitars with the age of middle age. Within the next membership we will try a lower price that associates with average.
* Rule 7 test:
  + Age = 42, model = custom, last price = $6000 (average)
  + Expectation: return a price within the bounds of expensive
  + Result:
    - Fair value price: $ 44,577.26 which is expensive
    - Well, my predictions are a bit off. Now, I’m thinking the reason it changed into the positive direction is because we wanted age to be the most determining factor. Since we increased the age by 4 years, I believe that can warrant a bit of an increase, so let’s set the age back down to 38 years old and see what happens.
      * Fair value price: $ 43,443.40
      * This time our prediction was correct. I think now we can clearly say that $43,443.40 is the max price if the guitar has a maximum fuzzy value of 1 for the middle aged category.
* Rule 8 test:
  + Age = 45, model = standard, last price = $2500 (cheap)
  + Expectation: return a price within the bounds of average
  + Result:
    - Fair value price: $ 6,169.45 which is average
* Rule 9 test:
  + Age = 35, model = pro, last price = $1004 (cheap)
  + Expectation: return a price within the bounds of average
  + Result:
    - Fair value price: $ 0 which is not correct. Let’s see if there is a bug within the code. It appears I forgot to add the OR operation and did not include the cheap price level. Now that I have debugged the problem let’s run the experiment again.
    - Fair value price: $ 6,131.59 which is average!
* Rule 10 test:
  + Age = 40, model = custom, last price = $2000 (cheap)
  + Expectation: return a price within the bounds of average
  + Result:
    - Fair value price: $ 6,131.59 which is average
* Rule 11 test:
  + Age = 0, model = standard, last price = $14000 (expensive)
    - Note that for a guitar with age 0 and price $14000 most likely would not exist, this is solely to test the rule
  + Expectation: return a price within the bounds of average
  + Result:
    - Fair value price: $ 6,131.59 which is average
* Rule 12 test:
  + Age = 10, model = pro, last price = $9000 (expensive)
  + Expectation: return a price within the bounds of average
  + Result:
    - Fair value price: $ 6,177.23 which is average
* Rule 13 test:
  + Age = 17, model = custom, last price = $11000 (expensive)
  + Expectation: return a price within the bounds of expensive
  + Result:
    - Fair value price: $ 45,180.07 which is expensive
* Rule 14 test:
  + Age = 5, model = standard, last price = $2342 (expensive)
  + Expectation: return a price within the bounds of cheap
  + Result:
    - Fair value price: $ 1,761.91 which is cheap
    - This individual bought a depreciating asset :(
* Rule 15 test:
  + Age = 16, model = pro, last price = $2890 (expensive)
  + Expectation: return a price within the bounds of cheap
  + Result:
    - Fair value price: $ 1,784.51 which is cheap
* Rule 16 test:
  + Age = 13, model = custom, last price = $6000 (expensive)
  + Expectation: return a price within the bounds of average
  + Result:
    - Fair value price: $ 6,131.58 which is average

It appears that our fuzzy system has done what we expected it to do, although it seems a bit strange that some of our results are the same.

**Conclusion:**

All in all, we have gone over the basics of fuzzy sets within our project description. We also discussed the code flow of our project and how our trading and guitar script accessed our fuzzy logic system library. Lastly, we discussed in depth how our trading system operated on a low level and did some experiments within. We have determined that the fuzzy logic system did what we expected it to do within our trading indicator. But we still have yet to test the validity of the strategy within the market. In the future if we wanted this trading indicator to be used in industry, we would need to test it with some back tests. We would go in depth with using this indicator in a real strategy to determine whether or not this indicator can be used by itself to execute winning trades. We also could feed the results of this indicator as one of many nodes into a neural network. This would allow that trading system to use our indicator to determine if they should long or short, but the system won’t solely rely on the success of this indicator. Finally, we discussed the guitar fair market analyzer at a low level. We explained what effects the price of a guitar and then set up our fuzzy system with these metrics in mind. After that we conducted experiments to see if the fuzzy system was working properly. There were some results that gave us the same answer and we were able to determine that there are maximum prices that could be assigned based on if the age was within the maximum fuzzy value for said membership category. We also discovered a logic error within the code that was then fixed. It was determined that the guitar fair market value analyzer was working as expected. In the future I would combine some of the rules for standard and pro models as they had similar price action and I could have written a bit less code. I would also in the future try and hook this up to a web application that way someone could use it that way they don’t make a bad financial decision with selling their guitars.