Log Book

Science Fair Logbook

Date: October 9, 2025

Time Spent: 30min

Today's focus: Think of a rough science fair idea

Reasoning for focus:
- To start my project I need an idea to base it on

What I accomplished:
- Attempting to get a rough idea on what I want to do

Decisions I made:
- Revolve around Algorithmic Trading
- Choose an experimental project
- Compare the investment management of bulge bracket banks and fintech firms algorithmic trading of stock price volatility during periods of high market stress

Reasoning for decisions:
- Algorithmic trading fuses my interests tech and finance together
- Comparison projects provide me a greater insight towards a field as it focuses on a specific topic on multiple deviations rather than something singular brought by innovation and research based projects
- Comparing investment management techniques can help one understand which strategy tends to be more resistant in periods of economic downturn as that is what tends to hurt the average investor's portfolio the greatest

Conclusion:
- Fintech firms keep their algorithmic ideologies a secret so unviable
- The outcome is already known (Since algorithmic trading can quickly change their stock allocations and hold strategies based on smaller quick-time investments, it tends to do much better in these periods).

Date: October 10, 2025

Time Spent: 30min

Today's focus: Think of a rough science fair idea

Reasoning for focus:
- To start my project, I need an idea to base it on

What I accomplished:
- Another attempt at getting a rough idea of what I want to do

Decisions I made:
- This time, compare versatile algorithmic strategies to traditional long-term ones in high periods of stress

Reasoning for decisions:
- Makes my project more viable (fintech firms keep their ideologies a secret)
- Enables me to gain knowledge across multiple algorithmic trading strategies

Conclusion:
- Unspecific and needs to be further refined

Date: October 14, 2025

Time Spent: 30min

Today's focus: Think of a rough science fair idea

Reasoning for focus:
- To start my project I need an idea to base it on

What I accomplished:
- Another attempt at getting a rough idea on what I want to do

Decisions I made:
- Creating a more tailored question
- Using a specific period of economic downturn(COVID19: Specifically February to May 2020)
- Using specific metrics such a maximum drawdown

Reasoning for decisions:
- Makes my project more comprehensive
- Enables me to gain knowledge across multiple algorithmic trading strategies
- The COVID19 downturn is considered a recent occurrence
- Maximum drawdown is a common metric utilized during market volatility

Conclusion:
- Became unsatisfied with idea
- Doesn't provide insight to a current problem

Date: October 16, 2025

Time Spent: 30min

Today's focus: Think of a rough science fair idea

Reasoning for focus:
- To start my project I need an idea to base it on

What I accomplished:
- Attempting to get a rough idea on what I want to do

Decisions I made:
- Research general problems brought by AI
- Apply Xai to Algorithmic trading

Reasoning for decisions:
- Algorithmic trading has advanced quite greatly creating a steep learning curve for individuals who want to explore the field

Conclusion:
- Satisfied with idea
- Provides insight to a current problem

Date: October 22, 2025

Time Spent: 1h30min

Today's focus: Research on financial ratios

Reasoning for focus:
- Ties into my role for another extracurricular

What I accomplished:
- Define what a Financial Ratio is
- Complete research on liquidity ratios
- Start research on profitability ratios

Decisions I made:
- Planned to cover 5 financial ratio groups (liquidity, profitability, price, efficiency and leverage)

Reasoning for decisions:
- Although financial ratios aren't a fool-proof way of evaluating a company's status, it can provide an overall idea
- Can integrate them in my Xai and use it to explain results

Conclusion:
- Use tomorrow to continue this research

Date: October 23, 2025

Time Spent: 4h

Today's focus: Continue research on financial ratios

Reasoning for focus:
- Ties into my role for another extracurricular

What I accomplished:
- Complete research on profitability ratios
- Complete research on efficiency ratios
- Complete research on leverage ratios
- Complete research on price ratios

Decisions I made:
- Planned to cover 5 financial ratio groups (liquidity, profitability, price, efficiency and leverage)
- Now decided to cover ratios based off risk (Risk Adjusted Return Ratios)

Reasoning for decisions:
- Although financial ratios aren't a fool-proof way of evaluating a company's status, it can provide an overall idea
- Can integrate them in my Xai and use it to explain results

Conclusion:
- Use tomorrow to continue this research

Date: October 24, 2025

Time Spent: 2h

Today's focus: Continue research on financial ratios + learn how to use numpy, pandas yfinance, and matplotlib

Reasoning for focus:

- To start my project, I need an idea to base it on

What I accomplished:

- Finished Risk Adjusted Return Ratios
- Created a simple plot using data to create monthly closed prices for individual stocks

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import yfinance as yf




def main():
  Ticker_Groups, full_data = compile()


  close_data = metricCalc().closing_prices_calc(full_data)
  metricVisuals().visual_close(close_data)




def compile():
  #type(rets) ----> Known as "Series": Is one dimensional
  # Using a dataframe which is two-dimensional
  Ticker_Groups={
      "The_Swim": ["AGG", "BSV", "SCHD"],
      "The_Bike": ["VOO", "MSFT", "CEG", "WCN", "AMT"],
      "The_Run": ["QQQ", "ICLN", "PAVE", "NVDA", "AVAV"]
      }


  all_tickers = [ticker for group in Ticker_Groups.values() for ticker in group]
```

```
  print("Downloading data...")
  rets = yf.download(
  tickers = all_tickers,
  interval="1mo",
  start = "2015-01-01"
  )


  #rets.dropna(inplace=True) ---> Most recnt stock started trading 2022
  # Price Series
  rets = rets.to_period("M")


  print("Data successfully compiled.")
  return Ticker_Groups, rets


class metricCalc:
  def closing_prices_calc(self, full_data):
      close_prices = full_data["Close"]


      return close_prices


class metricVisuals:
  def visual_close(self, close_data): #the reason for self is bc it directs phython to refer to the
instance of the class
      print("Creating plot...")
      plt.figure(figsize=(10, 5)) # makes a new canvas
      close_data.plot(title="Monthly Adjusted Close Prices") # Corrected variable


      # Add necessary labels and display the plot
      plt.ylim(0, close_data.max().max() * 1.05) # Sets the max Y-limit to 5% above the highest value
      plt.xlabel("Date")
      plt.ylabel("Adjusted Close Price ($)")
      plt.legend(title="Ticker")
      plt.show()
      print("Plot sucsessfully created.")
```

```
if __name__ == "__main__":
  main()
```

Decisions I made:
- Use python for simplicity

Reasoning for decisions:
- Can integrate them in my Xai and use it to explain results

Conclusion:
- Start another aspect of the project for further initial research
- Advance my basic coding structure

Date: October 25, 2025

Time Spent: 1.5h

Today's focus: refine my coding skills in terms of grouping and overall code complexity

Reasoning for focus:
- To further practice producing stock metrics

What I accomplished:
- Added to the monthly adjusted close prices by creating a new graph grouping the stocks together

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import yfinance as yf




def main():
  Ticker_Groups, full_data = compile()
```

```
  mc = metricCalc() #scalc stands for stock calculations
  mv = metricVisuals() #stock visuals


  #FOR INDIVIDUAL GRAPHS
  single_close_data = mc.single_closing_prices_calc(full_data)
  mv.single_visual_close(single_close_data)


  group_close_data =  mc.group_closing_prices_calc(full_data, Ticker_Groups)
  mv.group_visual_close(group_close_data)


  plt.show()


def compile():
  #INPUT
  #type(rets) ----> Known as "Series": Is one dimensional
  # Using a dataframe which is two-dimensional
  Ticker_Groups={
      "the_swim_avg": ["AGG", "BSV", "SCHD"],
      "the_bike_avg": ["VOO", "MSFT", "CEG", "WCN", "AMT"],
      "the_run_avg": ["QQQ", "ICLN", "PAVE", "NVDA", "AVAV"]
      }


  all_tickers = [ticker for group in Ticker_Groups.values() for ticker in group]


  print("Downloading data...")
  rets = yf.download(
  tickers = all_tickers,
  interval="1mo",
  start = "2015-01-01"
  )


  #rets.dropna(inplace=True) ---> Most recnt stock started trading 2022
  # Price Series
  rets = rets.to_period("M")
```

```
  print(rets.tail()) # Mini view of chlov


  print("Data sucsesfullly compiled.")
  return Ticker_Groups, rets


class metricCalc:
  #PROCESSING
  def single_closing_prices_calc(self, full_data):
      single_close_prices = full_data["Close"]


      return single_close_prices


  def group_closing_prices_calc(self, full_data, Ticker_Groups):
      # Getting data
      all_close_prices = full_data["Close"]


      #Seperating data
      the_swim_close_prices = Ticker_Groups["the_swim_avg"]
      the_bike_close_prices = Ticker_Groups["the_bike_avg"]
      the_run_close_prices = Ticker_Groups["the_run_avg"]


      #finding averages within groups
      the_swim_close_avg = all_close_prices[ the_swim_close_prices].mean(axis = 1) # an axis of 1
means the average is compiled horizontaly (the column)
      the_bike_close_avg = all_close_prices[ the_bike_close_prices].mean(axis = 1) # ^--- gives me
single ticker value per date
      the_run_close_avg =  all_close_prices[ the_run_close_prices].mean(axis = 1)


      # Compile under a single value + rename
      group_close_prices = pd.DataFrame({ #####
          "The Swim Average": the_swim_close_avg,
          "The Bike Average": the_bike_close_avg,
```

```
          "The Run Average": the_run_close_avg
      })



      return group_close_prices



class metricVisuals:
  #OUTPUT
  def single_visual_close(self, single_close_data): #the reason for self is bc it directs phython to
refer to the instance of the class
      print("Creating plot for individual tickers...")
      # ----------> plt.figure(figsize=(10, 5)) # makes a new canvas

      #FOR INDIVIDUAL GRAPHS
      single_close_data.plot(title="Monthly Adjusted Close Prices (Single)") # Corrected variable



      # Add necessary labels and display the plot
      plt.ylim(0, single_close_data.max().max() * 1.05) # Sets the max Y-limit to 5% above the highest
value
      plt.xlabel("Date")
      plt.ylabel("Adjusted Close Price ($)")
      plt.legend(title="Ticker")
      print("Plot sucsessfully created.")



  def group_visual_close(self, group_close_data):
      print("Creating plot for ticker groups...")
      # ---------->plt.figure(figsize=(10, 5)) # makes a new canvas



      #FOR INDIVIDUAL GRAPH
      group_close_data.plot(title="Monthly Adjusted Close Prices for Groups") # Corrected variable



      # Add necessary labels and display the plot
      plt.ylim(0, group_close_data.max().max() * 1.05) # Sets the max Y-limit to 5% above the highest
value
      plt.xlabel("Date")
      plt.ylabel("Adjusted Close Price ($)")
```

```
      plt.legend(title="Ticker Group")
      print("Plot sucsessfully created.")



if __name__ == "__main__":
  main()
```

Decisions I made:

- Continue to pursue monthly adjusted closing prices

Reasoning for decisions:

- Adding a small complexity (creating groups) from something I already know helps me grow with stability

Conclusion:

- Learn how to use subplots

Date: November 1st, 2025

Time Spent: 2h

Today's focus: refine my coding skills in terms of plot management

Reasoning for focus:

- To create a neater visual

What I accomplished:

- Connected by graphs into a singular image

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import yfinance as yf




def main():
```

```
  Ticker_Groups, full_data = compile()


  mc = metricCalc() #scalc stands for stock calculations
  mv = metricVisuals() #stock visuals


  #FOR INDIVIDUAL GRAPHS
  single_close_data = mc.single_closing_prices_calc(full_data)
  mv.single_visual_close(single_close_data)


  group_close_data =  mc.group_closing_prices_calc(full_data, Ticker_Groups)
  mv.group_visual_close(group_close_data)


  plt.show()


def compile():
  #INPUT
  #type(rets) ----> Known as "Series": Is one dimensional
  # Using a dataframe which is two-dimensional
  Ticker_Groups={
      "the_swim_avg": ["AGG", "BSV", "SCHD"],
      "the_bike_avg": ["VOO", "MSFT", "CEG", "WCN", "AMT"],
      "the_run_avg": ["QQQ", "ICLN", "PAVE", "NVDA", "AVAV"]
      }


  all_tickers = [ticker for group in Ticker_Groups.values() for ticker in group]


  print("Downloading data...")
  rets = yf.download(
  tickers = all_tickers,
  interval="1mo",
  start = "2015-01-01"
  )
```

```
  #rets.dropna(inplace=True) ---> Most recnt stock started trading 2022
  # Price Series
  rets = rets.to_period("M")


  print(rets.tail()) # Mini view of chlov


  print("Data sucsesfullly compiled.")
  return Ticker_Groups, rets


class metricCalc:
  #PROCESSING
  def single_closing_prices_calc(self, full_data):
      single_close_prices = full_data["Close"]


      return single_close_prices


  def group_closing_prices_calc(self, full_data, Ticker_Groups):
      # Getting data
      all_close_prices = full_data["Close"]


      #Seperating data
      the_swim_close_prices = Ticker_Groups["the_swim_avg"]
      the_bike_close_prices = Ticker_Groups["the_bike_avg"]
      the_run_close_prices = Ticker_Groups["the_run_avg"]


      #finding averages within groups
      the_swim_close_avg = all_close_prices[ the_swim_close_prices].mean(axis = 1) # an axis of 1
means the average is compiled horizontaly (the column)
      the_bike_close_avg = all_close_prices[ the_bike_close_prices].mean(axis = 1) # ^--- gives me
single ticker value per date
      the_run_close_avg =  all_close_prices[ the_run_close_prices].mean(axis = 1)


      # Compile under a single value + rename
```

```
      group_close_prices = pd.DataFrame({ #####
          "The Swim Average": the_swim_close_avg,
          "The Bike Average": the_bike_close_avg,
          "The Run Average": the_run_close_avg
      })


      return group_close_prices


class metricVisuals:
  #OUTPUT
  def single_visual_close(self, single_close_data): #the reason for self is bc it directs phython to
refer to the instance of the class
      print("Creating plot for individual tickers...")
      # ----------> plt.figure(figsize=(10, 5)) # makes a new canvas

      #FOR INDIVIDUAL GRAPHS
      single_close_data.plot(title="Monthly Adjusted Close Prices (Single)") # Corrected variable


      # Add necessary labels and display the plot
      plt.ylim(0, single_close_data.max().max() * 1.05) # Sets the max Y-limit to 5% above the highest
value
      plt.xlabel("Date")
      plt.ylabel("Adjusted Close Price ($)")
      plt.legend(title="Ticker")
      print("Plot sucsessfully created.")


  def group_visual_close(self, group_close_data):
      print("Creating plot for ticker groups...")
      # ---------->plt.figure(figsize=(10, 5)) # makes a new canvas


      #FOR INDIVIDUAL GRAPH
      group_close_data.plot(title="Monthly Adjusted Close Prices for Groups") # Corrected variable


      # Add necessary labels and display the plot
      plt.ylim(0, group_close_data.max().max() * 1.05) # Sets the max Y-limit to 5% above the highest
```

```
value
      plt.xlabel("Date")
      plt.ylabel("Adjusted Close Price ($)")
      plt.legend(title="Ticker Group")
      print("Plot sucsessfully created.")



if __name__ == "__main__":
  main()
```

Decisions I made:
n/a

Reasoning for decisions:

- Adding a small complexity (creating groups) from something I already know helps me grow with stability

Conclusion:

- Add adjusted standard deviation metric

Date: November 5th, 2025

Time Spent: 1.5h

Today's focus: Add adjusted standard deviation metric

Reasoning for focus:

- To create a more comprehensive visual

What I accomplished:

- Finished figuring out the basics of these libraries

```
import pandas as pd # manipulating stocks
import numpy as np #part of pandas
import matplotlib.pyplot as plt #visual aid
import yfinance as yf #stock info from yahoo finance
```

```
def main():
  #1) Bridging the classes + objects together
  Ticker_Groups, full_data = compile() #


  mc = metricCalc() #scalc stands for stock calculations
  mv = metricVisuals() #stock visuals


  single_close_data = mc.single_closing_prices_calc(full_data)
  group_close_data =  mc.group_closing_prices_calc(full_data, Ticker_Groups)


  single_return_data = mc.single_return_prices_calc(single_close_data)
  group_return_data = mc.group_return_prices_calc(group_close_data)


  single_std_data = mc.single_std_calc(single_return_data)
  group_std_data = mc.group_std_calc(group_return_data)


  mv.combined_visual_close(single_close_data, group_close_data, single_return_data, group_return_data,
single_std_data, group_std_data)


  #Displaying graph
  plt.show()


def compile():
  #INPUT
  #type(full_data) ----> Known as "Series": Is one dimensional
  # Using a dataframe which is two-dimensional


  #prep for group_closing_prices_calc
  Ticker_Groups={
      #Creating series within an object
      "The Swim Average": ["AGG", "BSV", "SCHD"],
      "The Bike Average": ["VOO", "MSFT", "CEG", "WCN", "AMT"],
      "The Run Average": ["QQQ", "ICLN", "PAVE", "NVDA", "AVAV"]
```

```
    }


  #compile separate groups + prep for full_data
  all_tickers = [ticker for group in Ticker_Groups.values() for ticker in group]


  #full_data = shorthand for returns ---> In this case shorthand returning semi-filtered data
  print("Downloading data...")
  try:
      full_data = yf.download( #Calling library
      tickers = all_tickers, #Finding stock info
      interval="1mo", #Compile to ideal indice
      start = "2015-01-01" #approx. 10 years
  )
  except Exception as e:
      print("Error downloading data.", e)
      return None, None


  #full_data.dropna(inplace=True) ---> Most recent stock started trading after 2015, cuts info before
that off
  # Price Series
  full_data = full_data.to_period("M") #Consistent decimals


  print(full_data.tail()) # Mini view of chlov


  print("Data successfullly compiled.")
  return Ticker_Groups, full_data


class metricCalc:
  #PROCESSING


  def single_closing_prices_calc(self, full_data):
      single_close_prices = full_data["Close"] #Get close values from yf
```

```
    return single_close_prices


def group_closing_prices_calc(self, full_data, Ticker_Groups):

    #Getting data
    all_close_prices = full_data["Close"]


    #Dynamic array
    #Memory: CD-AC


    group_close_prices = {} #Creating dictionary


    #Loop statement
    for group_closing, tickers in Ticker_Groups.items():
        avg_prices = all_close_prices[tickers].mean(axis=1) #get averages
        group_close_prices[group_closing]= avg_prices #column name


    return pd.DataFrame(group_close_prices)


def single_return_prices_calc(self, single_close_data):
    single_return_prices = single_close_data.pct_change()# Percent change from last row


    return single_return_prices


def group_return_prices_calc(self, group_close_data):
    group_return_prices = group_close_data.pct_change()# Percent change from last row


    return group_return_prices


def single_std_calc(self, single_return_data): #index by year
    df = single_return_data.copy() #create new DataFrame
```

```
        single_std = df.groupby(df.index.year).std() #makes "Year" the index


        return single_std


    def group_std_calc(self, group_return_data): #index by year
        df = group_return_data.copy() #create new DataFrame


        group_std = df.groupby(df.index.year).std() #makes "Year" the index


        return group_std


class metricVisuals:
    #OUTPUT


    #Figure = Entire canvas --> shorthand = fig
    #Axes = single plot --> shorthand = axes
    def single_visual_close(self, single_close_data, ax): #the reason for self is bc it directs phython to refer to the instance of the class
        print("Creating plot for Monthly Adjusted Close Prices (Single)...")

        single_close_data.plot(ax=ax, title="Monthly Adjusted Close Prices (Single)") # Corrected variable


        # Add necessary labels and display the plot
        ax.set_ylim(0, single_close_data.max().max() * 1.05) # Sets the max Y-limit to 5% above the highest value
        ax.set_xlabel("Date")
        ax.set_ylabel("Adjusted Close Price ($)")
        ax.legend(title="Ticker")
        print("Plot successfully created.")
```

```
  def group_visual_close(self, group_close_data, ax):
      print("Creating plot for Monthly Adjusted Close Prices for Groups...")
       group_close_data.plot(ax=ax, title="Monthly Adjusted Close Prices for Groups") # Corrected
variable


      # Add necessary labels and display the plot
      ax.set_ylim(0, group_close_data.max().max() * 1.05) # Sets the max Y-limit to 5% above the
highest value
      ax.set_xlabel("Date")
      ax.set_ylabel("Adjusted Close Price ($)")
      ax.legend(title="Ticker Group")
      print("Plot successfully created.")


  def single_visual_return(self, single_return_data, ax):
      print("Creating plot for Monthly Adjusted Return Prices (Single)...")

      single_return_data.plot(ax=ax, title="Monthly Adjusted Return Prices (Single)") # Corrected
variable


      # Add necessary labels and display the plot
      ax.set_ylim(single_return_data.min().min() * 1.05, single_return_data.max().max() * 1.05)
      ax.set_xlabel("Date")
      ax.set_ylabel("Adjusted Return Price ($)")
      ax.legend(title="Ticker")
      print("Plot successfully created.")


  def group_visual_return(self, group_return_data, ax):
      print("Creating plot for Monthly Adjusted Return Prices for Groups...")
       group_return_data.plot(ax=ax, title="Monthly Adjusted Return Prices for Groups") # Corrected
variable


      # Add necessary labels and display the plot
      ax.set_ylim(group_return_data.min().min() * 1.05, group_return_data.max().max() * 1.05)
      ax.set_xlabel("Date")
      ax.set_ylabel("Adjusted Return Price ($)")
      ax.legend(title="Ticker Group")
```

```
        print("Plot successfully created.")


    def single_visual_std(self, single_std_data, ax):
        print("Creating plot for Yearly Adjusted Standard Deviaiton (Single)...")

        single_std_data.plot(ax=ax, title="Yearly Adjusted Standard Deviaiton (Single)") # Corrected
variable


        # Set the x-axis to represent years (the index of the DataFrame)
        ax.set_xticks(single_std_data.index)  # The x-ticks will be the years
        ax.set_xticklabels(single_std_data.index.astype(str))  # Set the tick labels to the years as
strings




        # Add necessary labels and display the plot
        ax.set_ylim(single_std_data.min().min() * 1.05, single_std_data.max().max() * 1.05)
        ax.set_xlabel("Date")
        ax.set_ylabel("Adjusted Standard Deviaiton")
        ax.legend(title="Ticker")
        print("Plot successfully created.")


    def group_visual_std(self, group_std_data, ax):
        print("Creating plot for Yearly Adjusted Standard Deviation for Ticker Groups...")

        group_std_data.plot(ax=ax, title="Yearly Adjusted Standard Deviation for Ticker Groups") #
Corrected variable


        # Set the x-axis to represent years (the index of the DataFrame)
        ax.set_xticks(group_std_data.index)  # The x-ticks will be the years
        ax.set_xticklabels(group_std_data.index.astype(str))  # Set the tick labels to the years as
strings
```

```
      # Add necessary labels and display the plot
      ax.set_ylim(group_std_data.min().min() * 1.05, group_std_data.max().max() * 1.05)
      ax.set_xlabel("Date")
      ax.set_ylabel("Adjusted Standard Deviaiton")
      ax.legend(title="Ticker")
      print("Plot successfully created.")


  def combined_visual_close(self, single_close_data, group_close_data, single_return_data,
group_return_data, single_std_data, group_std_data):
      fig, axes = plt.subplots(
          nrows = 2,
          ncols = 3,
          figsize= (35, 10)
          )


      # Adjust spacing between plots
      fig.subplots_adjust(left=0.044,
                          bottom=0.058,
                          right=0.096,
                          top=0.964,
                          wspace=0.193,
                          hspace=0.193)


      #There are 4 axes (subplots) in total. In coding we start from 0.
      self.single_visual_close(single_close_data, ax=axes[0,0])
      self.group_visual_close(group_close_data, ax=axes[1,0])


      self.single_visual_return(single_return_data, ax=axes[0,1])
      self.group_visual_return(group_return_data, ax=axes[1,1])


      self.single_visual_std(single_std_data, ax=axes[0,2])
      self.group_visual_std(group_std_data, ax=axes[1,2])


      plt.tight_layout() #Autofits graphs to prevent overlapping words
      print("Plots successfully combined.")
```

```
if __name__ == "__main__": #Performs the code
  main()
```

Decisions I made:
- To take a break from coding and continue research

Reasoning for decisions:
- Create a string backbone on what im going to code and why it aids my project

Conclusion:
- Start researching about types of trading algorithms

Break Explanation

The Alberta Teachers Association's strike had been ended by the government (bill 2) and school started again. I had to resume school and catch up with the schedule, At this time my school work became too heavy of a workload to regularly pursue my extracurriculars

Date: December 23, 2025

Time Spent: 10min

Today's focus: Work on a little bit of my project.

Reasoning for focus:
- To maintain at least some continuity towards the project

What I accomplished:
- I started researching trading algorithms

Decisions I made:
- I chose to do the TWAP, VWAP, Simple Moving Average Crossover, Bollinger Bands, Breakout trading

Reasoning for decisions:

- They are the most popular type of trading algorithms

Conclusion:
- Even though I only finished one portion, I still accomplished something

Date: January 23, 2025

Time Spent: 2h

Today's focus: Finish trading algorithms research

Reasoning for focus:
- To catch up on my project once again and make up for lost time while catching up with schoolwork (was out of the country for 1.5 weeks near the end of semester 1).

What I accomplished:
- I finished researching trading algorithms

Decisions I made:
- Choosing bollinger band mean reversion strategy
- Choosing to devote my long weekend to the project

Reasoning for decisions:
- Was best for my risk stratified indexes and the strategy is easy to integrate with the libraries I utilized (numpy, pandas, matplotlib, yfinance)

Conclusion:
- Decided to research about different types of xAI tomorrow

Date: January 24, 2026

Time Spent: 1.6h

Today's focus: Start & finish xAI research

Reasoning for focus:
- To ensure a strong outline of what im going to end up coding

What I accomplished:
- I almost finished researching xAI applications (one portion remaining)

Decisions I made:

- Choosing to research SHAP, LIME Permutation feature importance, Grad-CAM, ELI5

Reasoning for decisions:

- FInding diverse xAI applications to investigate with extension will best suit the project

Conclusion:

- Decided to research about different types of xAI tomorrow

Date: January 25, 2026

Time Spent: 45min

Today's focus: Finish xAI research

Reasoning for focus:

- To ensure a strong outline of what im going to end up coding

What I accomplished:

- Finished all xAI research

Decisions I made:

- Choosing to research SHAP, LIME Permutation feature importance, Grad-CAM, ELI5

Reasoning for decisions:

- FInding diverse xAI applications to investigate with extension will best suit the project

Conclusion:

- Settled on ELI5 & SHAP

Date: January 26, 2026

Time Spent: 2h

Today's focus: Creating a methodology based off information & reaching out to my teacher about officially signing up for cysf

Reasoning for focus:

- To ensure a strong outline of what im going to end up coding

What I accomplished:

- Finished my methodology
- Since my school had a flex day, student who did not receive an invitation to visit school and catch up on classwork are not invited to meet up with a teacher
- I could not fill out anything on the website (Project Info) because I was not connected with the school yet despite this I began to write a brief descriptions about my project on a separate doc and email my teacher access

Decisions I made:
- To further research the HCI impact behind my project and transform my project to encompass meaningful psychological analysis rather than plain preference (e.g. overtrust)
- Created skeleton sections onto my project regarding the main issues my project addresses (will fill out a detailed description later, right now its in point form)

Reasoning for decisions:
- Add further depth behind my project & create a clear guide to follow

Conclusion:
- Finish the remainder of today's work tomorrow

Date: January 27, 2026

Time Spent: 2h

Today's focus: Finish brief descriptions and start coding again

Reasoning for focus:
- To ensure a strong outline of what im going to end up coding

What I accomplished:
- Finished my brief description and associated human risks (classified my project as low risk)
- Attempted to set up my environment (failed)

Decisions I made:
- Attempt to restart my VSCode environment on a day where I have more time

Reasoning for decisions:
- I became agitated with my broken environment after attempting to fix it for 1h+
- felt like my frustration wasn't optimal to continue my work

Conclusion:
- Continue my work on the weekend with a clear mind

Date: January 28, 2026

Time Spent: 25min

Today's focus: Meet up with my teacher at school today and ask about being added to the platform

Reasoning for focus:
- To ensure that my project can be approved before the deadline

What I accomplished:
- Talking to my teacher

Decisions I made:
- n/a
- My teacher told me to check with him on Thursday (when science fair club was taking place so he can add me more conveniently)

Reasoning for decisions:
- n/a

Conclusion:
- Come to Sci fair club on thursday during lunchtime

Date: January 29, 2026

Time Spent: 25min

Today's focus: Get officially added onto the cysf platform

Reasoning for focus:
- To ensure that can participate in cysf

What I accomplished:
- Got added

Decisions I made:
- To copy and paste my project description and 2a work into the cysf sections

Reasoning for decisions:
- To provide the decision making community with ample time to review my work

Conclusion:

- Finally finished my official intro application to participate

Date: January 30, 2026

Time Spent: 1.5h

Today's focus: Set up VSCode so I can code again

Reasoning for focus:
- So I can expand on my previous work

What I accomplished:
- Setting up VSCode
- Planned out how I am going to transform my plotting data to contain a Bollinger bands strategy

Decisions I made:
- Figure out what parts of my work that I should delete (e.g. no longer useful or redundant, like my plotting for standard deviation)

Reasoning for decisions:
- Decluttering unnecessary elements can keep my code clear and concise

Conclusion:
- Finally, extend my work tomorrow

Date: January 31, 2026

Time Spent: 3h

Today's focus: Create a Bollinger bands mean reversion strategy using historical data

Reasoning for focus:
- To create material to be utilized in my survey

What I accomplished:
- Finished creating my strategy
- I had an issue with my mouse and searched up results online to no avail, so I used an LLM to provide specific help. My automatic player also became broken, however, I couldnt fix this one so from now on I am running my code manually by going into terminal and typing python3 and pasting my application name into it

```
import pandas as pd # manipulating stocks
import numpy as np #part of pandas
import matplotlib.pyplot as plt #visual aid
import yfinance as yf #stock info from yahoo finance


def main():

  #1) Bridging the classes + objects together
  Ticker_Groups, full_data = compile() #


  mc = metricCalc() #scalc stands for stock calculations
  mv = metricVisuals() #stock visuals


  single_close_data = mc.single_closing_prices_calc(full_data)
  group_close_data =  mc.group_closing_prices_calc(full_data, Ticker_Groups)


  single_return_data = mc.single_return_prices_calc(single_close_data)
  group_return_data = mc.group_return_prices_calc(group_close_data)

  bb_mid, bb_upper, bb_lower = mc.bollinger_bands_calc(group_close_data)
  percent_b, band_width = mc.bollinger_features_calc(group_close_data, bb_mid, bb_upper, bb_lower)
  signals = mc.mean_reversion_signal(percent_b)

  mv.combined_visual_close(single_close_data, group_close_data, single_return_data, group_return_data,
bb_mid, bb_upper, bb_lower, signals)

  print(signals.tail())


  #Displaying graph
  plt.show()


def compile():
  #INPUT
  #type(full_data) ----> Known as "Series": Is one dimensional
  # Using a dataframe which is two-dimensional


  #prep for group_closing_prices_calc
  Ticker_Groups={
     #Creating series within an object
```

```
    "The Swim Average": ["AGG", "BSV", "SCHD"],
    "The Bike Average": ["VOO", "MSFT", "CEG", "WCN", "AMT"],
    "The Run Average": ["QQQ", "ICLN", "PAVE", "NVDA", "AVAV"]
    }


  #compile separate groups + prep for full_data
  all_tickers = [ticker for group in Ticker_Groups.values() for ticker in group]


  #full_data = shorthand for returns ---> In this case shorthand returning semi-filtered data
  print("Downloading data...")
  try:
    raw_data = yf.download( #Calling library
    tickers = all_tickers, #Finding stock info
    interval="1mo", #Compile to ideal indice
    start = "2015-01-01" #approx. 10 years
  )
    full_data = raw_data["Close"]

  except Exception as e:
    print("Error downloading data.", e)
    return None, None


  #full_data.dropna(inplace=True) ---> Most recent stock started trading after 2015, cuts info before that off
  # Price Series
  full_data = full_data.ffill().dropna()

  full_data.index = full_data.index.tz_localize(None)


  print(full_data.tail()) # Mini view of chlov
  print("Data successfullly compiled.")
  return Ticker_Groups, full_data


class metricCalc:
  #PROCESSING


  def single_closing_prices_calc(self, full_data):
    return full_data


  def group_closing_prices_calc(self, full_data, Ticker_Groups):
```

```
    #Getting data
    all_close_prices = full_data


    #Dynamic array
    #Memory: CD-AC


    group_close_prices = {} #Creating dictionary


    #Loop statement
    for group_closing, tickers in Ticker_Groups.items():
        avg_prices = all_close_prices[tickers].mean(axis=1) #get averages
        group_close_prices[group_closing]= avg_prices #column name


    return pd.DataFrame(group_close_prices)


def single_return_prices_calc(self, single_close_data):
    single_return_prices = single_close_data.pct_change()# Percent change from last row


    return single_return_prices

def group_return_prices_calc(self, group_close_data):
    group_return_prices = group_close_data.pct_change()# Percent change from last row


    return group_return_prices

def bollinger_bands_calc(self, group_close_data, window = 20, num_std = 2):
    rolling_mean = group_close_data.rolling(window).mean() #equilibrium
    rolling_std = group_close_data.rolling(window).std() #volatility

    upper_band = rolling_mean + num_std * rolling_std #Bands are the statistical boundary
    lower_band = rolling_mean - num_std * rolling_std

    return rolling_mean, upper_band, lower_band

def bollinger_features_calc(self, group_close_data, bb_mid, bb_upper, bb_lower):
    percent_b = (group_close_data - bb_lower) / (bb_upper - bb_lower) #%b for price location
    band_width = (bb_upper - bb_lower) / bb_mid
```

```
    return percent_b, band_width

  def mean_reversion_signal(self, percent_b):
    signals = pd.DataFrame(index = percent_b.index)

    # Creating a Buy/Sell signal for each group
    for col in percent_b.columns:
     s = percent_b[col] #This has become a series
     signals[f"{col}_Buy"] = s < 0.1 #f"{col}.. is short for format this tring with the varibale col on it
     signals[f"{col}_Sell"] = s > 0.9

    return signals

class metricVisuals:
  #OUTPUT


  #Figure = Entire canvas --> shorthand = fig
  #Axes = single plot --> shorthand = axes
  def single_visual_close(self, single_close_data, ax): #the reason for self is bc it directs phython to refer to the instance of the class
    print("Creating plot for Monthly Adjusted Close Prices (Single)...")

    single_close_data.dropna().plot(ax=ax, title="Monthly Adjusted Close Prices (Single)") # Corrected variable


    # Add necessary labels and display the plot
    ax.set_ylim(0, single_close_data.max().max() * 1.05) # Sets the max Y-limit to 5% above the highest value
    ax.set_xlabel("Date")
    ax.set_ylabel("Adjusted Close Price ($)")
    ax.legend(title="Ticker")
    print("Plot successfully created.")


  def group_visual_close(self, group_close_data, ax):
    print("Creating plot for Monthly Adjusted Close Prices for Groups...")

    group_close_data.dropna().plot(ax=ax, title="Monthly Adjusted Close Prices for Groups") # Corrected variable


    # Add necessary labels and display the plot
    ax.set_ylim(0, group_close_data.max().max() * 1.05) # Sets the max Y-limit to 5% above the highest value
    ax.set_xlabel("Date")
    ax.set_ylabel("Adjusted Close Price ($)")
```

```
        ax.legend(title="Ticker Group")
        print("Plot successfully created.")


    def single_visual_return(self, single_return_data, ax):
        print("Creating plot for Monthly Adjusted Return Prices (Single)...")

        single_return_data.plot(ax=ax, title="Monthly Adjusted Return Prices (Single)") # Corrected variable


        # Add necessary labels and display the plot
        ax.set_ylim(single_return_data.min().min() * 1.05, single_return_data.max().max() * 1.05)
        ax.set_xlabel("Date")
        ax.set_ylabel("Adjusted Return Price ($)")
        ax.legend(title="Ticker")
        print("Plot successfully created.")

    def group_visual_return(self, group_return_data, ax):
        print("Creating plot for Monthly Adjusted Return Prices for Groups...")

        group_return_data.plot(ax=ax, title="Monthly Adjusted Return Prices for Groups") # Corrected variable


        # Add necessary labels and display the plot
        ax.set_ylim(group_return_data.min().min() * 1.05, group_return_data.max().max() * 1.05)
        ax.set_xlabel("Date")
        ax.set_ylabel("Adjusted Return Price ($)")
        ax.legend(title="Ticker Group")
        print("Plot successfully created.")

    def combined_visual_close(self, single_close_data, group_close_data, single_return_data,
group_return_data, bb_mid, bb_upper, bb_lower, signals):
        fig, axes = plt.subplots(
            nrows = 2,
            ncols = 3,
            figsize= (35, 10)
            )


        # Adjust spacing between plots
        fig.subplots_adjust(left=0.044,
                            bottom=0.058,
                            right=0.096,
                            top=0.964,
                            wspace=0.193,
                            hspace=0.193)
```

```
        #There are 4 axes (subplots) in total. In coding we start from 0.
        self.single_visual_close(single_close_data, ax=axes[0,0])
        self.group_visual_close(group_close_data, ax=axes[1,0])

        self.single_visual_return(single_return_data, ax=axes[0,1])
        self.group_visual_return(group_return_data, ax=axes[1,1])

         #Bollinger overlay

        group_close_data.plot(ax=axes[0,2])
        bb_upper.plot(ax=axes[0,2], linestyle = "--", color = "red", label = "Upper Band")
        bb_lower.plot(ax=axes[0,2], linestyle = "--", color = "green", label = "Lower Band")

        for col in signals.columns[::2]: #Buy columns
            group_name = col.replace("_Buy", "").replace("Sell", "")
            axes[0,2].scatter(
                signals.index[signals[col]].values,
                group_close_data.loc[signals[col], group_name],
                marker = "^",
                color = "green",
                s = 100,
                label = f"{group_name} Buy"
                )

        for col in signals.columns[1::2]:
            group_name = col.replace("_Buy", "").replace("_Sell", "")
            axes[0,2].scatter(
                signals.index[signals[col]].values,
                group_close_data.loc[signals[col], group_name],
                marker = "v",
                color = "red",
                s = 100,
                label = f"{group_name} Sell"
                )

         #Clean up legend
        handles, labels = axes[0,2].get_legend_handles_labels()
        by_label = dict(zip(labels, handles))
        axes[0,2].legend(by_label.values(), by_label.keys(), title = "Signals")

        plt.tight_layout() #Autofits graphs to prevent overlapping words
        print("Plots successfully combined.")

if __name__ == "__main__": #Performs the code
```

```
  main()

    - Made it more aesthetic
import pandas as pd # manipulating stocks
import numpy as np #part of pandas
import matplotlib.pyplot as plt #visual aid
import yfinance as yf #stock info from yahoo finance


def main():

  #1) Bridging the classes + objects together
  Ticker_Groups, full_data = compile() #


  mc = metricCalc() #scalc stands for stock calculations
  mv = metricVisuals() #stock visuals


  single_close_data = mc.single_closing_prices_calc(full_data)
  group_close_data =  mc.group_closing_prices_calc(full_data, Ticker_Groups)


  single_return_data = mc.single_return_prices_calc(single_close_data)
  group_return_data = mc.group_return_prices_calc(group_close_data)

  bb_mid, bb_upper, bb_lower = mc.bollinger_bands_calc(group_close_data)
  percent_b, band_width = mc.bollinger_features_calc(group_close_data, bb_mid, bb_upper, bb_lower)
  rsi_data = mc.rsi_calc(group_close_data, period = 14)
  signals = mc.mean_reversion_signal(percent_b, rsi_data)

  mv.combined_visual_close(group_close_data, bb_mid, bb_upper, bb_lower, signals)

  print(signals.tail())


  #Displaying graph
  plt.show()


def compile():
  #INPUT
  #type(full_data) ----> Known as "Series": Is one dimensional
  # Using a dataframe which is two-dimensional
```

```
  #prep for group_closing_prices_calc
  Ticker_Groups={
      #Creating series within an object
      "The Swim Average": ["AGG", "BSV", "SCHD"],
      "The Bike Average": ["VOO", "MSFT", "CEG", "WCN", "AMT"],
      "The Run Average": ["QQQ", "ICLN", "PAVE", "NVDA", "AVAV"]
      }


  #compile separate groups + prep for full_data
  all_tickers = [ticker for group in Ticker_Groups.values() for ticker in group]


  #full_data = shorthand for returns ---> In this case shorthand returning semi-filtered data
  print("Downloading data...")
  try:
      raw_data = yf.download( #Calling library
      tickers = all_tickers, #Finding stock info
      interval="1wk", #Compile to ideal indice
      start = "2015-01-01" #approx. 10 years
  )
      full_data = raw_data["Close"]

  except Exception as e:
      print("Error downloading data.", e)
      return None, None


  #full_data.dropna(inplace=True) ---> Most recent stock started trading after 2015, cuts info before that off
  # Price Series
  full_data = full_data.ffill().dropna()

  full_data.index = full_data.index.tz_localize(None)


  print(full_data.tail()) # Mini view of chlov
  print("Data successfullly compiled.")
  return Ticker_Groups, full_data


class metricCalc:
  #PROCESSING


  def single_closing_prices_calc(self, full_data):
      return full_data
```

```
  def group_closing_prices_calc(self, full_data, Ticker_Groups):

      #Getting data
      all_close_prices = full_data


      #Dynamic array
      #Memory: CD-AC


      group_close_prices = {} #Creating dictionary


      #Loop statement
      for group_closing, tickers in Ticker_Groups.items():
          avg_prices = all_close_prices[tickers].mean(axis=1) #get averages
          group_close_prices[group_closing]= avg_prices #column name


      return pd.DataFrame(group_close_prices)


  def single_return_prices_calc(self, single_close_data):
      single_return_prices = single_close_data.pct_change()# Percent change from last row


      return single_return_prices

  def group_return_prices_calc(self, group_close_data):
      group_return_prices = group_close_data.pct_change()# Percent change from last row


      return group_return_prices

  def bollinger_bands_calc(self, group_close_data, window = 8, num_std = 1.5):
      rolling_mean = group_close_data.rolling(window).mean() #equilibrium
      rolling_std = group_close_data.rolling(window).std() #volatility

      upper_band = rolling_mean + num_std * rolling_std #Bands are the statistical boundary
      lower_band = rolling_mean - num_std * rolling_std

      return rolling_mean, upper_band, lower_band

  def bollinger_features_calc(self, group_close_data, bb_mid, bb_upper, bb_lower):
```

```
    percent_b = (group_close_data - bb_lower) / (bb_upper - bb_lower) #%b for price location
    band_width = (bb_upper - bb_lower) / bb_mid

    return percent_b, band_width

  def rsi_calc(self, group_close_data, period = 14):
    delta = group_close_data.diff()

    gain = delta.where(delta > 0, 0)
    loss = -delta.where(delta < 0, 0)

    avg_gain = gain.ewm(alpha = 1/period, min_periods = period, adjust = False).mean()
    avg_loss = loss.ewm(alpha = 1/period, min_periods = period, adjust = False).mean()

    rs = avg_gain / avg_loss
    rsi = 100 - (100 / (1 + rs))

    return rsi

  def mean_reversion_signal(self, percent_b, rsi):
    signals = pd.DataFrame(index = percent_b.index)

    # Creating a Buy/Sell signal for each group
    for col in percent_b.columns:
     s = percent_b[col] #This has become a series
     signals[f"{col}_Buy"] = (s < 0.3) & (rsi[col] < 30) #f"{col}.. is short for format this tring with the varibale col
on it
     signals[f"{col}_Sell"] = (s > 0.9)& (rsi[col] > 70)

    return signals

class metricVisuals:
  #OUTPUT


  #Figure = Entire canvas --> shorthand = fig
  #Axes = single plot --> shorthand = axes

   def combined_visual_close(self, group_close_data, bb_mid, bb_upper, bb_lower, signals):

     #Bollinger overlay
    plt.style.use("dark_background")
    fig, ax = plt.subplots(figsize=(12, 7))

    colors = {
    "The Swim Average": "white",
```

```
    "The Bike Average": "#FFFFE0",
    "The Run Average": "#ADD8E6"}

    for target in group_close_data.columns:
      ax.plot(
        group_close_data.index,
        group_close_data[target],
        label = target,
        linewidth = 2
        )

      ax.fill_between(
        group_close_data.index,
        bb_lower[target],
        bb_upper[target],
        alpha = 0.2
        )

      ax.plot(
        group_close_data.index,
        bb_upper[target],
        linewidth = 0.8,
        linestyle = "--",
        color = "red",
        alpha = 0.9
        )
      ax.plot(
        group_close_data.index,
        bb_lower[target],
        linewidth = 0.8,
        linestyle = "--",
        color = "red",
        alpha = 0.9
        )

    buy_col = f"{target}_Buy"
    sell_col = f"{target}_Sell"

    for col in signals.columns:
      group_name = col.replace("_Buy", "").replace("_Sell", "")
      idx = signals.index[signals[col]]

      if "_Buy" in col:
       ax.scatter(
       idx,
       group_close_data.loc[idx, group_name],
```

```
        marker = "^",
        facecolor = "#228b22",
        s = 180,
        edgecolor = "green",
        label = "Institutional Buy",
        zorder = 10
         )

       elif "_Sell" in col:
        group_name = col.replace("_Sell", "")
        ax.scatter(
        idx,
        group_close_data.loc[idx, group_name],
        marker = "v",
        facecolor = "red",
        s = 180,
        edgecolor = "#8b0000",
        label =  "Institutional Sell",
        zorder = 10
         )

      #Clean up legend
     ax.set_title(f"Triathlon Backtesting Stratgy: Bollinger Bands Mean Reversion + RSI", fontsize = 16)
     ax.set_ylabel(f"Adjusted Closing Price ($)")
     ax.grid(color = "gray", linestyle = ":", alpha = 0.5)

     handles, labels = ax.get_legend_handles_labels()
     by_label = dict(zip(labels, handles))
     ax.legend(by_label.values(), by_label.keys(), loc = "upper left", title = "Signals")

     plt.tight_layout() #Autofits graphs to prevent overlapping words
     print("Plots successfully created.")

if __name__ == "__main__": #Performs the code
  main()
```

Decisions I made:

- Decided to change my monthly time frame to weekly
- Decided to extend my strategy using RSI (Relative Strength Indicator), to ensure more accurate sections to buy and sell stocks
- Use the dark mode for my graphs

Reasoning for decisions:

- Made the graph appear more professional
- Increased my algorithm's efficiency

Conclusion:
- Create an interactive tool that makes it easier to read and navigate my graphs

Date: February 1, 2026

Time Spent: 6h+

Today's focus: Implement a reading tool in my Bollinger bands mean reversion + RSI strategy

Reasoning for focus:
- To create material to be utilized in my survey

What I accomplished:
- Finished creating my tool

NOTE** When implementing this tool it didn't move, so I asked a LLM for guidance and it told me to use the "TkAgg" function in matplotlib to make it work.

```
import pandas as pd # manipulating stocks
import numpy as np #part of pandas
import matplotlib
import matplotlib.pyplot as plt #visual aid
matplotlib.use("TkAgg") #forces an interactive window
import matplotlib.dates as mdates # internal date numbers
import yfinance as yf #stock info from yahoo finance

def main():
    # 1) Bridging the classes + objects together
    Ticker_Groups, full_data = compile()

    mc = metricCalc() #scalc stands for stock calculations
    mv = metricVisuals() #stock visuals

    single_close_data = mc.single_closing_prices_calc(full_data)
    group_close_data =  mc.group_closing_prices_calc(full_data, Ticker_Groups)

    single_return_data = mc.single_return_prices_calc(single_close_data)
    group_return_data = mc.group_return_prices_calc(group_close_data)

    bb_mid, bb_upper, bb_lower = mc.bollinger_bands_calc(group_close_data)
    percent_b, band_width = mc.bollinger_features_calc(group_close_data, bb_mid, bb_upper, bb_lower)
    rsi_data = mc.rsi_calc(group_close_data, period = 14)
    signals = mc.mean_reversion_signal(percent_b, rsi_data)

    mv.combined_visual_close(group_close_data, bb_mid, bb_upper, bb_lower, signals)
```

```
    print(signals.tail())

    # Displaying graph
    plt.show()

def compile():
    # prep for group_closing_prices_calc
    Ticker_Groups={
        "The Swim Average": ["AGG", "BSV", "SCHD"],
        "The Bike Average": ["VOO", "MSFT", "CEG", "WCN", "AMT"],
        "The Run Average": ["QQQ", "ICLN", "PAVE", "NVDA", "AVAV"]
    }

    all_tickers = [ticker for group in Ticker_Groups.values() for ticker in group]

    print("Downloading data...")
    try:
        raw_data = yf.download(
            tickers = all_tickers,
            interval="1wk",
            start = "2020-01-01"
        )
        full_data = raw_data["Close"]
    except Exception as e:
        print("Error downloading data.", e)
        return None, None

    full_data = full_data.bfill().ffill()
    full_data.index = full_data.index.tz_localize(None)

    print(full_data.tail())
    print("Data successfully compiled.")
    return Ticker_Groups, full_data

class metricCalc:
    # PROCESSING
    def single_closing_prices_calc(self, full_data):
        return full_data

    def group_closing_prices_calc(self, full_data, Ticker_Groups):
        all_close_prices = full_data
        group_close_prices = {}

        for group_closing, tickers in Ticker_Groups.items():
            avg_prices = all_close_prices[tickers].mean(axis=1) # get averages
```

```
            group_close_prices[group_closing]= avg_prices # column name

        return pd.DataFrame(group_close_prices)

    def single_return_prices_calc(self, single_close_data):
        return single_close_data.pct_change()

    def group_return_prices_calc(self, group_close_data):
        return group_close_data.pct_change()

    def bollinger_bands_calc(self, group_close_data, window = 8, num_std = 1.5):
        rolling_mean = group_close_data.rolling(window).mean()
        rolling_std = group_close_data.rolling(window).std()
        upper_band = rolling_mean + num_std * rolling_std
        lower_band = rolling_mean - num_std * rolling_std
        return rolling_mean, upper_band, lower_band

    def bollinger_features_calc(self, group_close_data, bb_mid, bb_upper, bb_lower):
        percent_b = (group_close_data - bb_lower) / (bb_upper - bb_lower)
        band_width = (bb_upper - bb_lower) / bb_mid
        return percent_b, band_width

    def rsi_calc(self, group_close_data, period = 14):
        delta = group_close_data.diff()
        gain = delta.where(delta > 0, 0)
        loss = -delta.where(delta < 0, 0)
        avg_gain = gain.ewm(alpha = 1/period, min_periods = period, adjust = False).mean()
        avg_loss = loss.ewm(alpha = 1/period, min_periods = period, adjust = False).mean()
        rs = avg_gain / avg_loss
        rsi = 100 - (100 / (1 + rs))
        return rsi

    def mean_reversion_signal(self, percent_b, rsi):
        signals = pd.DataFrame(index = percent_b.index)
        for col in percent_b.columns:
            s = percent_b[col]
            signals[f"{col}_Buy"] = (s < 0.3) & (rsi[col] < 30)
            signals[f"{col}_Sell"] = (s > 0.9) & (rsi[col] > 70)
        return signals

class metricVisuals:
    def combined_visual_close(self, group_close_data, bb_mid, bb_upper, bb_lower, signals):
        plt.style.use("dark_background")
        fig, ax = plt.subplots(figsize=(12, 7))

        colors = {"The Swim Average": "white", "The Bike Average": "#FFFFE0", "The Run Average": "#ADD8E6"}
```

```
    for target in group_close_data.columns:
        ax.plot(group_close_data.index, group_close_data[target], label = target, linewidth = 2)
        ax.fill_between(group_close_data.index, bb_lower[target], bb_upper[target], alpha = 0.3)
        # Drawing bands matched to asset color
        ax.plot(group_close_data.index, bb_upper[target], linewidth = 0.8, linestyle = "--", color = "#8B0000",
alpha = 0.5)
        ax.plot(group_close_data.index, bb_lower[target], linewidth = 0.8, linestyle = "--", color = "#228B22",
alpha = 0.5)

    for col in signals.columns:
        group_name = col.replace("_Buy", "").replace("_Sell", "")
        idx = signals.index[signals[col]]
        if "_Buy" in col:
            ax.scatter(idx, group_close_data.loc[idx, group_name], marker = "^", facecolor = "#228B22", s = 150,
edgecolor = "green", label = "Institutional Buy", zorder = 10)
        elif "_Sell" in col:
            ax.scatter(idx, group_close_data.loc[idx, group_name], marker = "v", facecolor = "red", s = 150,
edgecolor = "#8B0000", label = "Institutional Sell", zorder = 10)


    # --- SCANNER SETUP ---
    scan_line = ax.axvline(
        x = group_close_data.index[-1],
        color = "white", alpha = 0.8,
        linewidth = 2.0,
        zorder = 50
        )

    scan_text = ax.text(
        0.012, 0.79,
        "Move mouse to scan",
        transform = ax.transAxes,
        bbox = dict(facecolor = "black", alpha = 0.7),
        verticalalignment='top',
        zorder = 100,
        family = "monospace"
        )

    def mover(event):
        if event.inaxes == ax:
            x_val = event.xdata

            idx = mdates.date2num(group_close_data.index)
            distance = np.abs(idx - x_val)
            idx_num = np.argmin(distance)
```

```
            curr_date = group_close_data.index[idx_num]
            scan_line.set_xdata([curr_date])

            status = f"DATE: {curr_date.date()}\n" + "-"*20 + "\n"
            for group in group_close_data.columns:
                is_buy = signals.loc[curr_date, f"{group}_Buy"]
                is_sell = signals.loc[curr_date, f"{group}_Sell"]

                if is_buy:
                    msg = "BUY"
                elif is_sell:
                    msg = "SELL"
                else:
                    msg = "HOLD"

                status += f"{group:18}: {msg}\n"

            scan_text.set_text(status)
            fig.canvas.draw_idle()

        fig.canvas.mpl_connect("motion_notify_event", mover)

        ax.set_title("Triathlon Strategy: Bollinger + RSI Scanner", fontsize = 16)
        ax.set_ylabel("Adjusted Closing Price ($)")
        ax.grid(color = "gray", linestyle = ":", alpha = 0.5)

        # Clean Legend (Removes duplicates)
        handles, labels = ax.get_legend_handles_labels()
        by_label = dict(zip(labels, handles))
        leg = ax.legend(by_label.values(), by_label.keys(), loc = "upper left", framealpha = 0.7)
        leg.set_zorder(150)
        plt.tight_layout()

if __name__ == "__main__":
    main()
```

Decisions I made:

- Create a dynamic moving line

***Reason for long length: my environment broke and playbutton + pip did not work (NOTE: my playbutton is still broken which means that now i must run my code manually but i managed to fix my pip)

Reasoning for decisions:

- Made the graph appear more professional
- Increased my algorithms efficiency

Conclusion:

- Add SHAP applications next week

Date: February 6, 2026

Time Spent: 3.5h+

Today's focus: Refine my model and implement SHAP

Reasoning for focus:
- Advance to phase 3

What I accomplished:
- Attempted to refine for 2.5h+ and reduced model accuracy so deleted my prior work
- Watched videos about SHAP for an hour

Decisions I made:
- Delete my prior extension of a 40 week analysis because this made my algorithm too strict.

Reasoning for decisions:
- Was inefficient

Conclusion:
- Continue my SHAP research tomorrow

Date: February 7, 2026

Time Spent: 3.5h+

Today's focus: Finish SHAP implementation

Reasoning for focus:
- Advance to phase 3

What I accomplished:
- Implemented the SHAP Waterfall into my algorithm using an "on click" method

```
import pandas as pd # manipulating stocks
import numpy as np #part of pandas
import matplotlib
import matplotlib.pyplot as plt #visual aid
matplotlib.use("TkAgg") #forces an interactive window
```

```
import matplotlib.dates as mdates # internal date numbers
import yfinance as yf #stock info from yahoo finance
import shap #xAI extension
import tabulate

def main():
   # 1) Bridging the classes + objects together
   Ticker_Groups, full_data, fundamentals = compile()

   mc = metricCalc() #scalc stands for stock calculations
   xai = xAIProvider() #SHAP implementation
   mv = metricVisuals() #stock visuals

   single_close_data = mc.single_closing_prices_calc(full_data) #collects weekly closing prices for
each individual stock
   group_close_data =  mc.group_closing_prices_calc(full_data, Ticker_Groups) # collects weekly losing
prices for each risk-stratified index

   single_return_data = mc.single_return_prices_calc(single_close_data)
   group_return_data = mc.group_return_prices_calc(group_close_data)

   bb_mid, bb_upper, bb_lower = mc.bollinger_bands_calc(group_close_data)
   percent_b, band_width = mc.bollinger_features_calc(group_close_data, bb_mid, bb_upper, bb_lower)
   rsi_data = mc.rsi_calc(group_close_data, period = 14)
   signals = mc.mean_reversion_signal(percent_b, rsi_data)
   group_fundamentals = mc.calculate_group_fundamentals(Ticker_Groups, fundamentals)
   future_check = group_close_data.shift(-4) > group_close_data #did price go up after 4 weeks?
   buy_cols = [c for c in signals.columns if "_Buy" in c] #extract only buy signal results
   win_results = future_check.values[signals[buy_cols].values]
   win_results = win_results[~pd.isnull(win_results)] #ignores empty data point
   print(f"Algorithim Accuracy (Win Ratio): {np.mean(win_results):.2%}")

   mv.combined_visual_close(group_close_data, bb_mid, bb_upper, bb_lower, signals, xai, percent_b,
rsi_data, group_fundamentals)

   print(signals.tail())

   # Displaying graph
   plt.show()

def compile():
```

```
# prep for group_closing_prices_calc
Ticker_Groups={
    "The Swim Average": ["AGG", "BSV", "SCHD"],
    "The Bike Average": ["VOO", "MSFT", "CEG", "WCN", "AMT"],
    "The Run Average": ["QQQ", "ICLN", "PAVE", "NVDA", "AVAV"]
}

all_tickers = [ticker for group in Ticker_Groups.values() for ticker in group]

print("Downloading data...")
try:
    raw_data = yf.download(
        tickers = all_tickers,
        interval="1wk",
        start = "2019-11-01" #2 months earlier so datat can be filled
    )
    full_data = raw_data["Close"] # compiles only the section of data we need
except Exception as e:
    print("Error downloading data.", e)
    return None, None

full_data = full_data.bfill().ffill()
full_data.index = full_data.index.tz_localize(None)

print(full_data.tail())
print("Data successfully compiled.")

fundamentals = {}
for t in all_tickers:
    try:
        info = yf.Ticker(t).info
        #Debt-to-equity and NPM
        fundamentals[t] = {
            "D2E": info.get("debtToEquity", 0) / 100,
            "NPM": info.get("profitMargins", 0)
        }
    except:
        fundamentals[t] = {"D2E": 0, "NPM": 0}

return Ticker_Groups, full_data, fundamentals
```

```
class metricCalc:
   # PROCESSING
   #INPUT
   #type(rets) ----> Known as "Series": Is one dimensional
   # Using a dataframe which is two-dimensional
   def single_closing_prices_calc(self, full_data):
       return full_data

   def group_closing_prices_calc(self, full_data, Ticker_Groups):
       all_close_prices = full_data
       group_close_prices = {} #{} refers to the risk-stratified indexes

       for group_closing, tickers in Ticker_Groups.items():
           avg_prices = all_close_prices[tickers].mean(axis=1) # get averages
           group_close_prices[group_closing]= avg_prices # column name

       return pd.DataFrame(group_close_prices) #two-dimensional data management

   def single_return_prices_calc(self, single_close_data):
       return single_close_data.pct_change() #the percent aletration between the closing prices are
the return prices

   def group_return_prices_calc(self, group_close_data): #we already did the looping work in
group_close_data
       return group_close_data.pct_change()

   #lookback for 8wk, the short time good for algo trading -v       v----num_std refers to the width
of bollinger bands
   def bollinger_bands_calc(self, group_close_data, window = 8, num_std = 1.5): #developing our
startegy
       rolling_mean = group_close_data.rolling(window).mean() #get average
       rolling_std = group_close_data.rolling(window).std()#get standard deviation
       upper_band = rolling_mean + num_std * rolling_std
       lower_band = rolling_mean - num_std * rolling_std
       return rolling_mean, upper_band, lower_band

   def bollinger_features_calc(self, group_close_data, bb_mid, bb_upper, bb_lower):
       percent_b = (group_close_data - bb_lower) / (bb_upper - bb_lower) # tells exactly where price
is located in bands
       band_width = (bb_upper - bb_lower) / bb_mid #finds the range of bollinger bands to see
volatility
```

```
        return percent_b, band_width

    def rsi_calc(self, group_close_data, period = 14):
        delta = group_close_data.diff()
        gain = delta.where(delta > 0, 0) #delta checks price changes from one week to the next
        loss = -delta.where(delta < 0, 0)
        avg_gain = gain.ewm(alpha = 1/period, min_periods = period, adjust = False).mean()# modified
exponetial moving avergaes to smooth out price spikes
        avg_loss = loss.ewm(alpha = 1/period, min_periods = period, adjust = False).mean()
        rs = avg_gain / avg_loss
        rsi = 100 - (100 / (1 + rs))
        return rsi

    def mean_reversion_signal(self, percent_b, rsi): #relative strength index
        signals = pd.DataFrame(index = percent_b.index)
        for col in percent_b.columns:
            s = percent_b[col]
            signals[f"{col}_Buy"] = (s < 0.45) & (rsi[col] < 40) #buy if price is near bottom of band +
heavily oversold
            signals[f"{col}_Sell"] = (s > 0.85) & (rsi[col] > 65) #sell if price near top and is
undersold
        return signals

    def calculate_group_fundamentals(self, Ticker_Groups, fundamentals):
        group_stats = {}
        for group, tickers in Ticker_Groups.items():
            avg_d2e = np.mean([fundamentals[t]["D2E"] for t in tickers])
            avg_npm = np.mean([fundamentals[t]["NPM"] for t in tickers])
            group_stats[group] = {"Debt_Ratio": avg_d2e, "Profit_Margin": avg_npm}
        return group_stats

class xAIProvider:
    def internal_model_logic(self, data_array):
            #data_array[:, 0] = %B (positioning)
            #data_array[:, 1] = RSI (momentum)
            #data_array[:, 2] = Profit Margin (quality)
            #data_array[:, 3] = Debt Ratio (risking)

            positioning = (0.45 - data_array[:, 0])
            momentum = (40 - data_array[:, 1]) / 100
            quality = data_array[:, 2] * 2 # higher margin is better
```

```
            risk = -data_array[:, 3] #higher debt is worse

            return positioning + momentum + quality + risk

    def create_shap(self, pb_val, rsi_val, margin, debt, group_name, is_deceptive = False): #pb_val &
rsi_val are just placholders for now
        feature_row = np.array([[pb_val, rsi_val, margin, debt]])
        background = np.array([[0.5, 50, 0.1, 0.5]]) # netural starting point

        explainer = shap.KernelExplainer(self.internal_model_logic, background) #run SHAP
        shap_values = explainer.shap_values(feature_row)

        if is_deceptive: #switches logic around
            shap_values[0] = shap_values[0] * -1

        exp = shap.Explanation(
            values = shap_values[0],
            base_values = explainer.expected_value[0] if isinstance(explainer.expected_value,
np.ndarray) else explainer.expected_value,
            data = feature_row[0],
            feature_names = ["Positioning (%B)", "Market Momentum (RSI)", "Business Quality
(Margin)","Financial Risk (Debt)"]
        )

        plt.figure(figsize = (8, 4))
        shap.plots.waterfall(exp, show = False) #plot type
        plt.title(f"Visual-Quantitative (SHAP) - {group_name}")
        plt.tight_layout()

        plt.show(block = False)
        plt.pause(0.1)

class metricVisuals:
    def combined_visual_close(self, group_close_data, bb_mid, bb_upper, bb_lower, signals, xai,
percent_b, rsi_data, group_fundamentals):
        plt.style.use("dark_background") #adds professional look
        fig, ax = plt.subplots(figsize=(12, 7))

        colors = {"The Swim Average": "white", "The Bike Average": "#FFFFE0", "The Run Average":
"#ADD8E6"}
```

```
        for target in group_close_data.columns:
            ax.plot(group_close_data.index, group_close_data[target], label = target, linewidth = 2)
            """
            ax.fill_between(group_close_data.index, bb_lower[target], bb_upper[target], alpha = 0.3)
            # Drawing bands matched to asset color
            ax.plot(group_close_data.index, bb_upper[target], linewidth = 0.8, linestyle = "--", color
= "#8B0000", alpha = 0.5)
            ax.plot(group_close_data.index, bb_lower[target], linewidth = 0.8, linestyle = "--", color
= "#228B22", alpha = 0.5)
            """

        for col in signals.columns:
            group_name = col.replace("_Buy", "").replace("_Sell", "")
            idx = signals.index[signals[col]]
            if "_Buy" in col:
                ax.scatter(idx, group_close_data.loc[idx, group_name], marker = "^", facecolor =
"#228B22", s = 150, edgecolor = "green", label = "Institutional Buy", zorder = 10)
            elif "_Sell" in col:
                ax.scatter(idx, group_close_data.loc[idx, group_name], marker = "v", facecolor = "red",
s = 150, edgecolor = "#8B0000", label = "Institutional Sell", zorder = 10)
        ax.set_xlim(pd.Timestamp("2020-01-01"), group_close_data.index[-1]) #start graphing


        # --- SCANNER SETUP ---
        scan_line = ax.axvline(
            x = group_close_data.index[-1], #ensures that all indexes are checked
            color = "white", alpha = 0.8,
            linewidth = 2.0,
            zorder = 50
            )

        scan_text = ax.text(
            0.012, 0.79,
            "Move mouse to scan",
            transform = ax.transAxes,
            bbox = dict(facecolor = "black", alpha = 0.7),
            verticalalignment = 'top',
            horizontalalignment = "left",
            zorder = 100,
            family = "monospace"
            )
```

```
    def mover(event):
        if event.inaxes == ax: #checks if mouse on graph
            x_val = event.xdata #time

            idx = mdates.date2num(group_close_data.index) #dates become x-axis
            distance = np.abs(idx - x_val) #distance between mouse at point and every position
            idx_num = np.argmin(distance) #return index position
            curr_date = group_close_data.index[idx_num] # gets date
            scan_line.set_xdata([curr_date])

            status = f"DATE: {curr_date.date()}\n" + "-"*20 + "\n"
            for group in group_close_data.columns:
                is_buy = signals.loc[curr_date, f"{group}_Buy"]
                is_sell = signals.loc[curr_date, f"{group}_Sell"]

                if is_buy:
                    msg = "BUY"
                elif is_sell:
                    msg = "SELL"
                else:
                    msg = "HOLD"

                status += f"{group:18}: {msg}\n"

            scan_text.set_text(status)
            fig.canvas.draw_idle()

    def on_click(event):
        #SHAP extension
        if event.inaxes == ax: #checks if mouse on graph
            x_val = event.xdata #time

            idx = mdates.date2num(group_close_data.index) #dates become x-axis
            distance = np.abs(idx - x_val) #distance between mouse at point and every position
            idx_num = np.argmin(distance) #return index position
            curr_date = group_close_data.index[idx_num] # gets date

            y_val = event.ydata
            target = (np.abs(group_close_data.loc[curr_date] - y_val)).idxmin()
```

```
            p_b = percent_b.loc[curr_date, target] #loc = location
            rsi = rsi_data.loc[curr_date, target]
            margin = group_fundamentals[target]["Profit_Margin"]
            debt = group_fundamentals[target]["Debt_Ratio"]

            print(f"Generating SHAP for {curr_date.date()}...")
            xai.create_shap(p_b, rsi, margin, debt, target, is_deceptive = False)
            fig.canvas.draw_idle()

    fig.canvas.mpl_connect("motion_notify_event", mover) #shows algo choice
    fig.canvas.mpl_connect("button_press_event", on_click) #shows shap plot


    ax.set_title("Triathlon Strategy: Bollinger + RSI Scanner", fontsize = 16)
    ax.set_ylabel("Adjusted Closing Price ($)")
    ax.grid(color = "gray", linestyle = ":", alpha = 0.5)

    # Clean Legend (Removes duplicates)
    handles, labels = ax.get_legend_handles_labels()
    by_label = dict(zip(labels, handles))
    leg = ax.legend(by_label.values(), by_label.keys(), loc = "upper left", framealpha = 0.7)
    leg.set_zorder(150)
    plt.tight_layout()

    data = [ #glossary
        ["Financial Risk", "Debt-to-Equity", "Compares a company's total liabilities with its
shareholder equity. It is used to indicate the extent of a business's reliance on debt. a.k.a how much
they have vs. borrowed"],
        ["Buisness Quality", "Net Profit Margin", "Indicates the bottom line profit a business is
able to retain for each dollar of revenue earned."],
        ["Market Uncertainty", "Bollinger Band Width", "3 lines that encompass 95% of the stock and
indicates volatility (the smaller the width, the more stable)"],
        ["Positioning", "%B (price location)", "Where the stock is within the bollinger band. The
higher it is, the more oversold it is. The lower it is, the more underbought it is."]
    ]

    #headers
    headers = ["Cognitive Concept", "Ratio/Metric", "The 'Because' logic"]

    print("\n" + "="*80)
    print("GLOSSARY - TRIATHALON STRATEGY")
```

```
        print(tabulate.tabulate(data, headers = headers, tablefmt = "grid"))
        print("="*80 + "\n")

if __name__ == "__main__":
    main()
```

Decisions I made:

- Choose a waterfall plot for the best UX and presentation of information
- Begin ELI5 tommorrow

Reasoning for decisions:

- Implementing SHAP was a relatively short but updating my interactive system proved difficult

Conclusion:

- Start my ELI5 research tomorrow

Date: February 8, 2026

Time Spent: 5h

Today's focus: Implement ELI5 and finish survey creation

Reasoning for focus:

- Advance to Phase 4 and complete 3

What I accomplished:

- Emailed finished surveys for CYSF approval for consent based on completion
- Started ELI5
- Started survey creation for SHAP

Decisions I made:

- Include a glossary in both surveys to enable comprehension

Reasoning for decisions:

- To get ample data, I must start as soon as possible
- I plan to email some of my teachers, and if they agree, I would like to provide them with time to plan their week, accounting for this

Conclusion:

- I will be waiting for CYSF's response

Date: February 9, 2026

Time Spent: 15min

Today's focus: Review CYSF email + finish SHAP survey

Reasoning for focus:
- Advance to Phase 4 and complete 3

What I accomplished:
- I fully finalized a rough survey guideline around the assumption of consent by completion

Decisions I made:
- Fully complete the SHAP survey

Reasoning for decisions:
- Get closer to stage 4

Conclusion:
- I will be away for a week, but after I come back, all I have to do is complete the ELI5 implementation and survey creation

Date: February 21 & 22, 2026

NOTE*** Combined these dates because they both had the same purpose

Time Spent: 5h

Today's focus: Finish ELI5 + survey for ELI5

Reasoning for focus:
- Advance to Phase 4 and complete 3

What I accomplished:
- Finished ELI5

```
import pandas as pd # manipulating stocks
import numpy as np #part of pandas
import matplotlib
import matplotlib.pyplot as plt #visual aid
matplotlib.use("TkAgg") #forces an interactive window
import matplotlib.dates as mdates # internal date numbers
import yfinance as yf #stock info from yahoo finance
```

```
import shap #xAI extension
import tabulate #glossary

def main():
   # 1) Bridging the classes + objects together
   Ticker_Groups, full_data, fundamentals = compile()

   mc = metricCalc() #scalc stands for stock calculations
   xai = xAIProvider() #SHAP implementation
   mv = metricVisuals() #stock visuals

   single_close_data = mc.single_closing_prices_calc(full_data) #collects weekly closing prices for
each individual stock
   group_close_data =  mc.group_closing_prices_calc(full_data, Ticker_Groups) # collects weekly losing
prices for each risk-stratified index

   single_return_data = mc.single_return_prices_calc(single_close_data)
   group_return_data = mc.group_return_prices_calc(group_close_data)

   bb_mid, bb_upper, bb_lower = mc.bollinger_bands_calc(group_close_data)
   percent_b, band_width = mc.bollinger_features_calc(group_close_data, bb_mid, bb_upper, bb_lower)
   rsi_data = mc.rsi_calc(group_close_data, period = 14)
   signals = mc.mean_reversion_signal(percent_b, rsi_data)
   group_fundamentals = mc.calculate_group_fundamentals(Ticker_Groups, fundamentals)
   future_check = group_close_data.shift(-4) > group_close_data #did price go up after 4 weeks?
   buy_cols = [c for c in signals.columns if "_Buy" in c] #extract only buy signal results
   win_results = future_check.values[signals[buy_cols].values]
   win_results = win_results[~pd.isnull(win_results)] #ignores empty data point
   print(f"Algorithim Accuracy (Win Ratio): {np.mean(win_results):.2%}")

   mv.combined_visual_close(group_close_data, bb_mid, bb_upper, bb_lower, signals, xai, percent_b,
rsi_data, group_fundamentals)

   print(signals.tail())

   # Displaying graph
   plt.show()

def compile():
   # prep for group_closing_prices_calc
   Ticker_Groups={
```

```
        "The Swim Average": ["AGG", "BSV", "SCHD"],
        "The Bike Average": ["VOO", "MSFT", "CEG", "WCN", "AMT"],
        "The Run Average": ["QQQ", "ICLN", "PAVE", "NVDA", "AVAV"]
    }

    all_tickers = [ticker for group in Ticker_Groups.values() for ticker in group]

    print("Downloading data...")
    try:
        raw_data = yf.download(
            tickers = all_tickers,
            interval="1wk",
            start = "2019-11-01" #2 months earlier so datat can be filled
        )
        full_data = raw_data["Close"] # compiles only the section of data we need
    except Exception as e:
        print("Error downloading data.", e)
        return None, None

    full_data = full_data.bfill().ffill()
    full_data.index = full_data.index.tz_localize(None)

    print(full_data.tail())
    print("Data successfully compiled.")

    fundamentals = {}
    for t in all_tickers:
        try:
            info = yf.Ticker(t).info
            #Debt-to-equity and NPM
            fundamentals[t] = {
                "D2E": info.get("debtToEquity", 0) / 100,
                "NPM": info.get("profitMargins", 0)
            }
        except:
            fundamentals[t] = {"D2E": 0, "NPM": 0}

    return Ticker_Groups, full_data, fundamentals

class metricCalc:
    # PROCESSING
```

```
    #INPUT
    #type(rets) ----> Known as "Series": Is one dimensional
    # Using a dataframe which is two-dimensional
    def single_closing_prices_calc(self, full_data):
        return full_data

    def group_closing_prices_calc(self, full_data, Ticker_Groups):
        all_close_prices = full_data
        group_close_prices = {} #{} refers to the risk-stratified indexes

        for group_closing, tickers in Ticker_Groups.items():
            avg_prices = all_close_prices[tickers].mean(axis=1) # get averages
            group_close_prices[group_closing]= avg_prices # column name

        return pd.DataFrame(group_close_prices) #two-dimensional data management

    def single_return_prices_calc(self, single_close_data):
        return single_close_data.pct_change() #the percent aletration between the closing prices are
the return prices

    def group_return_prices_calc(self, group_close_data): #we already did the looping work in
group_close_data
        return group_close_data.pct_change()

    #lookback for 8wk, the short time good for algo trading -v       v----num_std refers to the width
of bollinger bands
    def bollinger_bands_calc(self, group_close_data, window = 8, num_std = 1.5): #developing our
startegy
        rolling_mean = group_close_data.rolling(window).mean() #get average
        rolling_std = group_close_data.rolling(window).std()#get standard deviation
        upper_band = rolling_mean + num_std * rolling_std
        lower_band = rolling_mean - num_std * rolling_std
        return rolling_mean, upper_band, lower_band

    def bollinger_features_calc(self, group_close_data, bb_mid, bb_upper, bb_lower):
        percent_b = (group_close_data - bb_lower) / (bb_upper - bb_lower) # tells exactly where price
is located in bands
        band_width = (bb_upper - bb_lower) / bb_mid #finds the range of bollinger bands to see
volatility
        return percent_b, band_width
```

```
   def rsi_calc(self, group_close_data, period = 14):
       delta = group_close_data.diff()
       gain = delta.where(delta > 0, 0) #delta checks price changes from one week to the next
       loss = -delta.where(delta < 0, 0)
       avg_gain = gain.ewm(alpha = 1/period, min_periods = period, adjust = False).mean()# modified
exponetial moving avergaes to smooth out price spikes
       avg_loss = loss.ewm(alpha = 1/period, min_periods = period, adjust = False).mean()
       rs = avg_gain / avg_loss
       rsi = 100 - (100 / (1 + rs))
       return rsi

   def mean_reversion_signal(self, percent_b, rsi): #relative strength index
       signals = pd.DataFrame(index = percent_b.index)
       for col in percent_b.columns:
           s = percent_b[col]
           signals[f"{col}_Buy"] = (s < 0.45) & (rsi[col] < 40) #buy if price is near bottom of band +
heavily oversold
           signals[f"{col}_Sell"] = (s > 0.85) & (rsi[col] > 65) #sell if price near top and is
undersold
       return signals

   def calculate_group_fundamentals(self, Ticker_Groups, fundamentals):
       group_stats = {}
       for group, tickers in Ticker_Groups.items():
           avg_d2e = np.mean([fundamentals[t]["D2E"] for t in tickers])
           avg_npm = np.mean([fundamentals[t]["NPM"] for t in tickers])
           group_stats[group] = {"Debt_Ratio": avg_d2e, "Profit_Margin": avg_npm}
       return group_stats

class xAIProvider:
   def internal_model_logic(self, data_array):
           #data_array[:, 0] = %B (positioning)
           #data_array[:, 1] = RSI (momentum)
           #data_array[:, 2] = Profit Margin (quality)
           #data_array[:, 3] = Debt Ratio (risking)

           positioning = (0.45 - data_array[:, 0])
           momentum = (40 - data_array[:, 1]) / 100
           quality = data_array[:, 2] * 2 # higher margin is better
           risk = -data_array[:, 3] #higher debt is worse
```

```
            return positioning + momentum + quality + risk

    def create_shap(self, pb_val, rsi_val, margin, debt, group_name, is_deceptive = True): #pb_val &
rsi_val are just placholders for now
        feature_row = np.array([[pb_val, rsi_val, margin, debt]])
        background = np.array([[0.5, 50, 0.1, 0.5]]) # netural starting point

        explainer = shap.KernelExplainer(self.internal_model_logic, background) #run SHAP
        shap_values = explainer.shap_values(feature_row)

        final_values = np.array(shap_values[0]) if isinstance(shap_values, list) else shap_values

        if is_deceptive: #switches logic around
            final_values = final_values * -1

        exp = shap.Explanation(
            values = shap_values[0],
            base_values = explainer.expected_value[0] if isinstance(explainer.expected_value,
np.ndarray) else explainer.expected_value,
            data = feature_row[0],
            feature_names = ["Positioning (%B)", "Market Momentum (RSI)", "Business Quality
(Margin)","Financial Risk (Debt)"]
        )

        return final_values.flatten()

    def create_eli5(self, shap_values, feature_names, group_name):
        reasons = {
            "Positioning (%B)": "where the price sits within the usual range",
            "Market Momentum (RSI)": "the current market conditons of the stock's valuation",
            "Business Quality (Margin)": "how much profit the company keeps from every dollar earned",
            "Financial Risk (Debt)": "the company's current debt levels"
        }
        explanation = f"ANALYSIS FOR: {group_name.upper()} \n"
        explanation += "The algorithim made the following desicion based on these factors: \n\n"
#sentence starter

        for i in range(len(feature_names)): #loop through list
            name = feature_names[i]
            val = shap_values[i]
```

```
            if abs(val) < 0.05: continue

            strength = "greatly " if abs(val) > 0.3 else ""
            direction = "improved" if val > 0 else "lowered"

            meaning = reasons.get(name, name)
            line = "- " + name + f" has {strength}{direction} the confidence by " + str(round(abs(val),
2)) + f" because of {meaning}" + "\n" #2 means the decimal value

            explanation = explanation + line

        return explanation

class metricVisuals:
    def combined_visual_close(self, group_close_data, bb_mid, bb_upper, bb_lower, signals, xai,
percent_b, rsi_data, group_fundamentals):
        plt.style.use("dark_background") #adds professional look
        fig, ax = plt.subplots(figsize=(12, 7))

        colors = {"The Swim Average": "white", "The Bike Average": "#FFFFE0", "The Run Average":
"#ADD8E6"}

        for target in group_close_data.columns:
            ax.plot(group_close_data.index, group_close_data[target], label = target, linewidth = 2)
            """
            ax.fill_between(group_close_data.index, bb_lower[target], bb_upper[target], alpha = 0.3)
            # Drawing bands matched to asset color
            ax.plot(group_close_data.index, bb_upper[target], linewidth = 0.8, linestyle = "--", color
= "#8B0000", alpha = 0.5)
            ax.plot(group_close_data.index, bb_lower[target], linewidth = 0.8, linestyle = "--", color
= "#228B22", alpha = 0.5)
            """

        for col in signals.columns:
            group_name = col.replace("_Buy", "").replace("_Sell", "")
            idx = signals.index[signals[col]]
            if "_Buy" in col:
                ax.scatter(idx, group_close_data.loc[idx, group_name], marker = "^", facecolor =
"#228B22", s = 150, edgecolor = "green", label = "Institutional Buy", zorder = 10)
            elif "_Sell" in col:
                ax.scatter(idx, group_close_data.loc[idx, group_name], marker = "v", facecolor = "red",
```

```
s = 150, edgecolor = "#8B0000", label = "Institutional Sell", zorder = 10)
      ax.set_xlim(pd.Timestamp("2020-01-01"), group_close_data.index[-1]) #start graphing



      # --- SCANNER SETUP ---
      scan_line = ax.axvline(
          x = group_close_data.index[-1], #ensures that all indexes are checked
          color = "white", alpha = 0.8,
          linewidth = 2.0,
          zorder = 50
          )

      scan_text = ax.text(
          0.012, 0.79,
          "Move mouse to scan",
          transform = ax.transAxes,
          bbox = dict(facecolor = "black", alpha = 0.7),
          verticalalignment = 'top',
          horizontalalignment = "left",
          zorder = 100,
          family = "monospace"
          )

      def mover(event):
          if event.inaxes == ax: #checks if mouse on graph
              x_val = event.xdata #time

              idx = mdates.date2num(group_close_data.index) #dates become x-axis
              distance = np.abs(idx - x_val) #distance between mouse at point and every position
              idx_num = np.argmin(distance) #return index position
              curr_date = group_close_data.index[idx_num] # gets date
              scan_line.set_xdata([curr_date])

              status = f"DATE: {curr_date.date()}\n" + "-"*20 + "\n"
              for group in group_close_data.columns:
                  is_buy = signals.loc[curr_date, f"{group}_Buy"]
                  is_sell = signals.loc[curr_date, f"{group}_Sell"]

                  if is_buy:
                      msg = "BUY"
                  elif is_sell:
```

```
                    msg = "SELL"
                else:
                    msg = "HOLD"

                status += f"{group:18}: {msg}\n"

            scan_text.set_text(status)
            fig.canvas.draw_idle()

        def on_click(event):
            #ELI5 extension
            if event.inaxes == ax:

                for t in list(ax.texts):
                    if t.get_zorder() == 200:
                        t.remove() # clear old box

                x_val = event.xdata #time
                idx = mdates.date2num(group_close_data.index) #dates become x-axis
                distance = np.abs(idx - x_val) #distance between mouse at point and every position
                idx_num = np.argmin(distance) #return index position
                curr_date = group_close_data.index[idx_num] # gets date

                y_val = event.ydata
                target = (np.abs(group_close_data.loc[curr_date] - y_val)).idxmin()

                p_b = percent_b.loc[curr_date, target] #loc = location
                rsi = rsi_data.loc[curr_date, target]
                margin = group_fundamentals[target]["Profit_Margin"]
                debt = group_fundamentals[target]["Debt_Ratio"]

                current_shap_values = xai.create_shap(p_b, rsi, margin, debt, target)
                narrative = xai.create_eli5(current_shap_values, ["Positioning (%B)", "Market Momentum
(RSI)", "Business Quality (Margin)","Financial Risk (Debt)"], target)

                ax.text(0.5, 0.5, narrative,
                    transform = ax.transAxes,
                    ha = "center", va = "center",
                    wrap = True, fontsize = 12,
                    zorder = 200,
                    color = "black",
```

```
                    bbox = {"facecolor": "white", "alpha": 0.9, "pad": 10 })

            print(f"Generating ELI5 for {target} on {curr_date.date()}...")
            fig.canvas.draw_idle()

        fig.canvas.mpl_connect("motion_notify_event", mover) #shows algo choice
        fig.canvas.mpl_connect("button_press_event", on_click) #shows shap plot



        ax.set_title("Triathlon Strategy: Bollinger + RSI Scanner", fontsize = 16)
        ax.set_ylabel("Adjusted Closing Price ($)")
        ax.grid(color = "gray", linestyle = ":", alpha = 0.5)

        # Clean Legend (Removes duplicates)
        handles, labels = ax.get_legend_handles_labels()
        by_label = dict(zip(labels, handles))
        leg = ax.legend(by_label.values(), by_label.keys(), loc = "upper left", framealpha = 0.7)
        leg.set_zorder(150)
        plt.tight_layout()

        data = [ #glossary
            ["Financial Risk", "Debt-to-Equity", "Compares a company's total liabilities with its
shareholder equity. It is used to indicate the extent of a business's reliance on debt. a.k.a how much
they have vs. borrowed"],
            ["Business Quality", "Net Profit Margin", "Indicates the bottom line profit a business can
retain for each dollar of revenue earned."],
            ["Market Uncertainty", "Bollinger Band Width", "3 lines that encompass 95% of the stock and
indicate volatility (the smaller the width, the more stable)"],
            ["Positioning", "%B (price location)", "Where the stock is within the Bollinger band. The
higher it is, the more oversold it is. The lower it is, the more underbought it is."]
        ]

        #headers
        headers = ["Cognitive Concept", "Ratio/Metric", "The 'Because' logic"]

        print("\n" + "="*80)
        print("GLOSSARY - TRIATHALON STRATEGY")
        print(tabulate.tabulate(data, headers = headers, tablefmt = "grid"))
        print("="*80 + "\n")
```

```
if __name__ == "__main__":
    main()
```

Decisions I made:
- Create the survey tomorrow

Reasoning for decisions:
- To further analyze code quality

Conclusion:
- Tomorrow, I will finish making the survey and email my teachers about possible class participation

Date: February 23, 2026

Time Spent: 2h

Today's focus: Finish survey for ELI5

Reasoning for focus:
- Advance to Phase 4 and complete 3

What I accomplished:
- Finished ELI5 survey
- Ask for parental suggestions and improvements; changed the wording on one of my questions to make it easier to understand
- Reached out to a few teachers

Decisions I made:
- Use the same dates to parallel the SHAP survey

Reasoning for decisions:
- To control the surveys as much as possible and add an extra layer of comparability

Conclusion:
- One of my teachers reached out to me so far and agreed to ask his class to participate tomorrow

Date: February 24, 2026

Time Spent: 10min

Today’s focus: Review incoming results + checked email

Reasoning for focus:
- Complete to Phase 4

What I accomplished:
- Read one teacher's response; he said he’ll see what he can do
- Looked at the results to calculate the overall difficulty of the survey

Decisions I made:
- Decided that the survey was at a good level of difficulty for the age group

Reasoning for decisions:
- n/a

Conclusion:
- Message my friends and wait for teacher responses

Date: February 25, 2026

Time Spent: 10min

Today’s focus: Review incoming results + checked email + messaged my friends

Reasoning for focus:
- Complete to Phase 4

What I accomplished:
- Sent messages to my friends

Decisions I made:
- Set a date limit for accepting responses by the end of 27th Feb

Reasoning for decisions:
- For ample time for data analysis

Conclusion:
- Await for responses

Date: February 26, 2026

Time Spent: 15min

Today's focus: Review incoming results + checked email

Reasoning for focus:
- Complete to Phase 4

What I accomplished:
- Got emails from 2 teachers, one said they'll post it up on D2L, and another asked if I can present my project to the class

Decisions I made:
- Visit the teacher's class who asked me to present after lunch
- Thanked the other teacher as well

Reasoning for decisions:
- Show my appreciation

Conclusion:
- Await for responses

Date: February 27, 2026

Time Spent: 4h+

Today's focus: Review incoming results + Begin filling out my CYSF research section

Reasoning for focus:
- Complete to Phase 4

What I accomplished:
- Began filling out my CYSF research section
- Almost finished the research section

Decisions I made:
- To finish my research section first
- To make project exclusively for Gen Z since other demographics did not contain ample information

Reasoning for decisions:
- It is the longest section

- Gives time for more people to do the survey

Conclusion:
- Complete research and begin other sections
- Closed the surveys at the end of the day

Date: February 28, 2026

Time Spent: 6h+

Today's focus: Continue filling out my CYSF research section

Reasoning for focus:
- Complete project before the deadline

What I accomplished:
- Almost filled the research section
- Filled out hypothesis
- Fillled out variables
- Start filling out procedures
- Added code blocks and images

NOTE*** My work from yesterday was deleted on the platform and I didnt get to save the verison, as a result I had to restart

Decisions I made:
- To finish my earlier section first

Reasoning for decisions:
- To comeback from yesterday's set back

Conclusion:
- Complete what I left off today tommorrow

Date: March 1, 2026

Time Spent: 6h+

Today's focus: Continue filling out my CYSF research section

Reasoning for focus:
- Complete project before the deadline

What I accomplished:
- FIlled out the research section
- Completed filling out the procedure section
- Started and finished working on observations

Decisions I made:
- To start analysis tommorrow

Reasoning for decisions:
- To research the proper terminology for correlations I found in obseravtions

Conclusion:
- Complete what I left off today tommorrow

Date: March 2, 2026

Time Spent: 6h+

Today's focus: Continue filling out my CYSF research section

Reasoning for focus:
- Complete project before the deadline

What I accomplished:
- Filled out analysis
- FIlled out conlcusions
- Filled out citations
- FIlled out applications
- Had problems with pasting stuff in properly so emailed CYSF and ask if I cqn use linked docs in instead
- Emailed some teachers to see if I can reveal their names in my acknowledgement section

Decisions I made:
- Double check logbook & assess quality of work tommorow

Reasoning for decisions:
- To ensure a quality project

Conclusion:
- Complete what I left off today tommorrow

- Wait back to hear from CYSF
- Start working on academia tomorrow
- If time persists include a video encompass the survey and interactive tools created

Date: March 3, 2026

Time Spent: 2h+

Today's focus: Review my Project

Reasoning for focus:
- Complete project before the deadline

What I accomplished:
- Finished review of my logbook
- Integrating a video of my tool in action
- Completing acknowledgements
- Completing attachments
- Completeing declarations

Decisions I made:
- Triple check work tonight and try to integrate a video if possible

Reasoning for decisions:
- To ensure a quality project

Conclusion:
- Fully finish CYSF project

Associated Research

Question
"Which explainable AI technique makes rule-based algorithmic trading strategies more transparent and understandable for non-experts?"

Refined:
“To what extent do visual-quantitative (SHAP) versus narrative-qualitative (ELI5) xAI explanations influence Gen Z user trust and mental model accuracy in rule-based algorithmic trading?”

Problem
What is the problem?
- AI Transparency
- AI Interpretability and explainability
- Tied into how they create stagnation, and it's getting harder to keep up
- The black box model

Why is it significant?
- Its harder to get into AI

What is already known?
- Xai
- Algorithmic Trading

What is the gap?
- Using Xai to make algorithmic trading more accessible
- Also mention how it corrodes trust and regulation alliance
- How human-level stagnation is often overshadowed by the dominant Xai focus on regulations
- Steep learning curve

Brief desc:
Algorithmic trading is often viewed as a "black box", fabricating a steep learning curve that prevents retail investors from accessing advanced strategies. This project investigates how different xAI techniques such a visual-quantitative (SHAP) vs. narrative-qualitative (ELI5) influence user trust and mental model accuracy.

Using python and multiple APIs (e.g. matplotlib, pandas, yfinance, numpy e.t.c), I developed a "Triathlon" portfolio by creating 3 risk-stratified indexes (Swim, Bike, Run) and applied it to a Bollinger Band mean-reversion strategy. I then conducted a human-subject study utilizing an initial “Within Subjects” model that encompasses a “black box” trade signal. Then I will split it into a “Between Subjects” model: Group A received SHAP bar charts, while Group B received an ELI5 natural language description.

By measuring the “Trust Delta” and using “What-If” logic tests, I will evaluate which method best improves logical comprehension. Furthermore, I incorporated some deceptive explanations to detect automation bias and overtrust. This explores the potential for the overtrust of xAI and AI. This research aims to find the optimal xAI framework to make algorithmic trading more transparent, accountable and accessible for non-experts.

Investing accessibility —> Financial Democratization

Recently, Artificial Intelligence(AI) and Financial Democratization movements have been taking the world of finance by storm. Becoming a key to solving inaccessibility in financial management across the middle class. Large language models (LLMs), like Gemini and ChatGPT, can now offer personalized financial advice for free, and the friendly UX behind budgeting apps encourages financial planning. According to Investopedia, “Firms like State Street and BlackRock have launched new exchange-traded funds (ETFs) to offer retail investors private credit strategies.” Could this be pivotal to solving the socioeconomic gap between classes, or a way for private companies capitalizing off of the FOMO of retail investors? How can we take AI to the next level for closing the gap between “high intelligence” and “low UX” within apps made for retail investing?

The ‘Black Box’ problem —> xAI’s role in AI transparency

Artificial Intelligence is an extremely powerful instrument, with the capabilities of detecting patterns that humans can’t simply recognize. Yet despite its potency, it cannot explain the reasoning behind its answers; it only reveals an input and output with a mystery in between. Laura Blattner, an assistant professor of finance at Stanford GSB, reflects on AI’s limitations when it comes to human-computer interaction(HCI), “If these black boxes are being used to make a high-stakes decision in lending, insurance, healthcare or the judicial system, we have to decide whether we feel comfortable not knowing exactly why the decision was being made.”

In the field of lending, an AI model can analyze over 600 variables that contribute to a credit score, over a few dozen that traditional loan decisions are derived from, for a more precise and fair answer. Yet despite this, U.S lenders tend to be skeptical of these tools due to their lack of credibility. When applying this to algorithmic trading, the aversion to implementing such tools in retail investing has been amplified. The lack of investment literacy and monetary sensitivity from non-experts, combined with the ominousness of AI, makes people skeptical.

Narrative vs. Visual comprehension—> Preference

Humans are often defined to be “visual” creatures, a survival trait passed down from our ancestors, but in this day and age, what format enables optimized comprehension? According to NIH, the high cognitive demand is that sequential logic from narratives is better suited for complex information. Yet for rapid processing and cognitive ease, visuals are key. Another interesting aspect of the study is that the extent of enhanced apprehension comes from exposure to such visuals; therefore, not universal.

In trading, for professionals, visual tools can clearly showcase an immediate result, while narrative tools can help traders find themes within patterns. When applying these factors into algorithmic trading for non-experts, it raises the question: What format (Narrative vs. Visual) helps beginner investors understand trading?

The ‘Overtrust’ Problem

“Overtrust” in terms of AI refers to the psychological phenomenon of human users placing excessive confidence in such systems, often overlooking critical analysis of AI’s input and reliability. This is often caused by multiple factors, with the most prominent one in algorithmic trading being the automation bias. The automation bias refers to the human tendency to trust automated systems when emotionally pressured, faced with complex information and lacking time. These are often aspects faced by non-experts/new traders.

2026 Adolescent exposure to AI

The rising generation is born in the “Digital Nativity" period, growing alongside social media, advanced UX and online learning during the COVID-19 pandemic. Artificial intelligence is relatively new, even for adolescents of this time, yet despite this, they have managed to integrate the use of artificial intelligence, mainly LLMs, into everyday life. Negatively impacting adolescent emotional, social and critical skills.

To what extent will Gen Z blindly trust AI and good UX?

Method

I. The Objective
To investigate which form of xAI a) visual-quantitative (SHAP) or b) narrative-qualitive (ELI5) best minimizes user discomfort and improves logical apprehension of rule-based algorithmic trading signals.

II. The Variables
Independent Variable:
The xAI model utilized for explanation (SHAP Bar Chart, or ELI5 Text).
+ Group A will then be shown an SHAP explanation
+ Group B will then be shown an ELI5 explanation

Dependant Variable:
1) Participant Trust Score (a scale of 1-10 in which 10 is the highest level of an individual's assurance with xAI output)
2) Mental Model Accuracy (score on the “What-If” logic questions).

Constant Variables:
1) Same Section 1 (Within-subject testing aspect of survey) for all participants
2) Same platform used to create each independent variable (VSCode)
3) Same glossary used with the same placement (in every page located at the bottom)
4) Same historical stock data within the same time frame(s)
5) Same Bollinger Band parameters
6) Same Risk-Stratified Index categories

III. The Procedure
Phase 1: Data Engineering & Index Stratification
I will segment multiple stocks into 3 distinct Risk-Stratified Index Models (the “Triathlon” Portfolio) from a well-rounded portfolio by grouping them up based on their market volatility:

+ Swim (Conservative): Low-volatility assets filtered for high Net Profit Margin (NPM) and low

Debt-to-Equity. They provide stability and slow growth to the portfolio.
+ Bike (Balanced): Moderate-volatility assets that provide standard risk and growth.
+ Run (Aggressive): High-volatility assets that are the pinnacle of “high-risk, high-reward”, when the markets are doing well, these stocks will propel this portfolio as they prioritize momentum and breakout potential.

Data will be sourced using the “yfinance” API, covering **5 months** of historical price action to ensure robust data sampling.

Phase 2: Algorithmic Logic (The “Heuristic Mapping”)
I will implement a Mean Reversion strategy using Bollinger Bands to act as the primary force behind the trading decisions.

+ Input Features (X): Bollinger Band %b (relative position), Band Width (Volatility), NPM, and Debt-to-Equity ratios.
+ The SHAP Wrapper: A conversion of my manual strategy into an automated model that xAI tools can interpret

Phase 3: xAI Implementation (The “Explanations”)
To make the algorithm transparent, I will implement 2 different systems:
1) Shapley Additive Explanations (SHAP): Its use of axiomatic attribution (mathematical intensity) to assign weight towards specific financial indicators can attribute to excellent visuals centered on game theory. (Summary Plots)
2) ELI5: I chose ELI5 to provide a natural language generator that translates the SHAP values into “simple English.” (e.g., “the AI is buying because…e.t.c) (Narrative Descriptions)

Phase 4: Human-Focused Applications and Evaluation(The “Delta Trust” Study)
The final phase involves a survey that starts with an Within-Subjects beginning to a Between-Subject’s split to analyze the effectiveness of xAI:

+ Baseline: All participants are shown a “Black Box” trade signal (Plain chart + Buy/Sell) and have access to a glossary at the bottom of each page
+ Diversion:
  i) Group A will then be shown an SHAP explanation accompanied with the same graph from the same time period
  ii) Group B will be shown the ELI5 explanation accompanied with the same graph from the same time period
+ Metrics: I will measure the “Trust Delta” (confidence boosted by xAI result) and “Mental Model Accuracy” (the participant’s ability to predict how the AI would react to a change in the stock’s data). This feature helps detect the prior abilities of the individual as well as their applications after being exposed to the model
+ Extra: Check for overtrust of xAI by including one wrong description(one without any implications of error) and see if it continues to boost or maintain confidence. Then I would implement another wrong description and this time describe a mismatch between the signal and the xAI.

Research

Initial Research: AI
What is artificial intelligence?
Artificial Intelligence utilizes computational learning and reading to carry out advanced tasks that need human intellect through computer science, data analytics, statistics, software engineering and neuroscience. An example of a complex assignment may be analyzing data, linguistic comprehension and insights.

Some Areas of AI
Machine Learning (ML)
- Systems that use data to analyze and make predictions or decisions without direct programming

Deep Learning (DL)
- ML subfield
- Uses many layers that require artificial neural networks
- More nuanced than ML
- Good at complex tasks like image and speech recognition

Natural Language Processing (NLP)
- Enables computers to understand, interpret and generate human language.
- Voice assistants, translation services and chatbots are powered by this

Computer Vision
- Allows computers to interpret visual information such as images and videos
- Used in facial recognition and self-driving cars

Initial Research: xAI
Note** The IML(Interpretable Model Learning) is also applicable to the topic as the field is relatively new and scientists as well as other professionals have yet to define the two terms. Generally an IML is reflective of a 3 step decision trees or models that are generally more interpretable to humans. Despite my project being relatively simple I would like to classify it under xAI because the financial applications and analysis of my work are aimed towards individuals with minimal knowledge of how markets work.

What is xAI?
Explainable artificial intelligence(Xai) is a set of processes and methods that enables human users to understand and trust the output and results produced by algorithms. Often describing the model in terms of objectives, expected impact and potential biases. It provides insight into model accuracy, fairness and transparency in decision making. It is a layer on top of an algorithm or AI that can trace and explain each decision.
xAI is the answer towards solving the blackbox problem.

Some types of xAI:
Local Interpretable Model-agnostic Explanations (LIME):
A common technique used to interpret black box models. It approximates explanations for a single case of the model's inputs or outputs. LIME explains the outcome generated for a specific AI input by modifying it to recognize its impact onto the outcome.

Shapley Additive Explanations (SHAP):
A technique that identifies the significance of each feature in a certain prediction. The concept is similar to game theory where it bases itself on the distribution of an award(the trade) fairly among players(the features determining the trigger of the trade).

Permutation feature importance:
The evaluation of each separate input feature to determine predictive power. This is done by generating a score for each feature by randomly shuffling its values while measuring prediction error. After observation, the next step is to measure the feature of choice through permutations(reshuffling). The importance score is derived by averaging the differences between the baseline and the reshuffled outcome.

Gradient-weighted Class Activation Mapping (Grad-CAM)
A model-specific method used to explain neural networks(highly advanced ai algorithms). For a paticu;ar input image(a graph in this case), the Grad-CAM combines the feature maps to decipher each part's relevance.

Explain it Like I'm 5 (ELI5) is an explainable tool that seeks to offer simple explanations for the outcome of models in a human-friendly manner. It provides an explanation considering relevant features of input that contribute to the final outcome of the model. This method is reliant on other xAI methods like SHAP, Permutation feature importance and LIME for explanation generation.

What Xai should I use?
I have decided to take a dual approach using SHAP and ELI5 to enable a comprehensive comparison of interpretability and test the improvement of trust by different outputs.

1) Shapley Additive Explanations (SHAP): I chose SHAP due to its axiomatic attribution (mathematical intensity) to assign weight to specific financial indicators. Its beginner-friendly visuals allow users to glance at a graph rooted in statistics to see which factor triggered the trade.
2) ELI5: I chose ELI5 to provide a natural language contrast. While SHAP is visual, this form of xAI is based purely off of simplification through solid descriptions. This allows me to test whether non-experts prefer graphical or narrative representation.

Initial Research: Trading Algorithms
What is algorithmic trading?
The use of computer algorithms to automate the trading process and execute these trades that specifical intervals by combining programming and financial market analysis. It is also known as automated trading or black-box trading (the issue we are trying to solve).

Pros:

- Fast execution during ideal time frames for the best possible price.
- Reduced transaction costs
- Elimination of human error: Bases trades purely off strategy and is not swayed by emotions like traditional(human) traders
- Backtesting: Using historical data to compare accuracy with real-time data to determine efficiency

Cons:

- Unpredictability in unforeseen future market disruptions: Being trained on historical data means that an algorithm cannot adapt to "Black Swan" events.
- Technological issues can disrupt the trading process and cause losses
- Speculation of increasing market volatility: Larger trading algorithms are known for significantly altering market prices.
- Lack of human judgment: The market is based on human demand and supply, in which mathematical models and historical data may help predict these choices, but will lack the true emotional essence behind the qualitative factors that define such movement (sentiment analysis (online media) may help, but the algorithm may remain restricted).

Types of Algorithmic Trading (Popular):
Simple Arbitrage (A common implementation within a strategy):
Buying a stock(often liquid) at a lower price and selling it at a higher price for monetary gain(often small) using price differentials in a different market (e.g. NYSE & LSE). The stocks utilized often depict market inefficiency(When the price of an asset does not reflect its true value).
NOTE: With the rise of technological efficiencies (stocks from different markets matching up more often), successful arbitrage has become harder to execute.

TWAP: Time Weighted Average Price Algorithm
The splitting of a large trading order into equal portions over a specific period prevents volatility caused by the price manipulation(stock inflation) of large trades.

$$\text{TWAP} = \frac{\sum_{i=1}^{n} P_i}{n}$$

Pi = Price at each interval (e.g. 5min)
n = Number of intervals

Pros:
- Price stability reduces the risk of being front-run by other algorithms

Cons:
- It is predictable, which means that it can underperform trends and is not suitable for high volatility.

VWAP: Volume Weighted Average Price Algorithm
The breaking up of a large order that takes both price and trading value into account, slicing the order to match the day's average price minimizing market impact. The execution aims to match or beat the daily VWAP buying below the indicator(bearish) and selling above(bullish).

$$VWAP = \frac{\sum_{1}^{n}(price \times volume)}{\sum_{1}^{n} volume}$$

n = Number of Intervals

Pros:
- Perfect for day traders due to the VWAP's intraday trend predictions
- Representative of a realistic equilibrium price
- Best to combine with other indicators for maximum success

Cons:
- Lagging indicator: Based on past data so may not be prepared for intraday volatility

Simple Moving Average Crossover: The next 2 below demonstrate a singular indicator and 2 trading philosophies. Similar to the VWAP, but are more rigid as it doesn't take volume into account placing equal emphasis on each point. This metric is great to see a rough trend of the stock's direction. It takes historical data and compares the stock to the average set across a length of time (e.g. 50-day SMA, 100-day SMA).

Mean Reversion (Bollinger Bands): Best SMA for choppy markets that bounces between an equilibrium
Bollinger Bands consist of 3 lines: the middle band is the moving average, with the upper and lower bands representative of a certain number of standard deviations above and below the mean. In a normal distribution about 95% of the values will be contained between these lines. By taking advantage of this principle and market elasticity, the algorithm will go short when it hits the upper band and long when it hits the lower band.
Mean=Sum of prices /Number of Observations

$$\text{Standard Deviation} = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \overline{x})^2}{n-1}}$$

where:

$x_i = \text{Value of the } i^{th} \text{ point in the data set}$

$\overline{x} = \text{The mean value of the data set}$

$n = \text{The number of data points in the data set}$

Pros:

- Representative of volatility measurement
- Clear identification of trends and potential breakpoints
- Excellent when used alongside other indicators

Cons:

- Lagging indicator

Breakout Trading: Best for markets where stocks have clear momentum
The breakout trading strategy involves entering positions when a stock moves past a defined boundary signalling the potential of a strong price trend. Investors go long when the stock “breaks” past the resistance level(upper) and short when it “breaks” below the support(lower) level. The more times the stock has touched these areas and the longer the stock remains within these boundaries, the better the outcome when the stock finally breaks out.

ATR(Average True Range)
The support and resistance levels are determined by Entry±(1.5×ATR)

What algorithm did I end up choosing and why?
A Mean Reversion strategy using Bollinger Bands applied to my "Triathlon" portfolio (Swim, Bike, and Run sections).

1) Interpretability: Bollinger Bands are excellent for Heuristic Mapping (conceptualization) due to their visual nature and association with oscillation. The use of a common statistical limit with a simple ideology (Standard deviation) is a topic that can be intuitively understood by non-experts.
2) Multidimensionality: Unlike simpler strategies, Bollinger Bands allow the integration of volatility and relative position through band width. This provides a nuanced data set that is closer to capturing the essence of the “Black Box” problem.
3) Risk Management: My Triathlon approach can further be highlighted by Mean Reversion by highlighting the contrasts and ceiling between each Risk-Stratified Index Model created to demonstrate the simplification of a well-rounded portfolio

RSI confluence
A technical indicator in momentum trading is a tool used to measure the speed and magnitude of a recent price change in a stock. It provides technical traders with information about whether market conditions are bullish (on the rise) or bearish (falling). An asset is considered overbought when above 70 and oversold at 30. By combining it with my Bollinger bands strategy, I decided to loosen the boundaries to 40 & 65 through hyperparameter tuning

to replicate the high frequency of algorithmic trading. This, paired with my %b threshold signals of 0.45 for buying (triggers when 5% below 8 week average) and 0. 65 for selling (15% above 8 week average) creates a well-constructed algorithm. The positive skew is targeted towards the bullish market and rising stock valuations during these periods.

Initial Research: Ratios & Metrics (p^2el^2)

What are Financial Ratios:
Through the analysis of financial statements (the main ones being the income statement, the balance sheet, and sometimes the cash flow statement). It is the comparison of one line-item against another. They are typically shown as a percentage.

What does Liquidity mean?
Liquidity refers to the ease at which something can be converted into cash.

What are Liquidity ratios?
They depict whether a business can cover its short-term debt obligations. The components to a liquidity ratio are located on the balance sheet. They can be shown as a number(most common) or a percentage.

What is conservative investing?
The prioritization of the preservation of capital with a weaker regard to growth or market returns. Think of stability.

The least conservative liquidity ratio: Current Ratio
Current Assets / Current Liabilities
What is the ideal amount for the Current Ratio?
Anything over a ratio of 1 is generally considered good as it means the company can pay off their debts.
A ratio under 1 signifies that the company may struggle to pay its short-term debt obligations.
Finally, a ratio higher than 2 may be considered excellent as they can cover their debt twice over. However, depending on which angle one looks at a ratio higher than 3 can potentially signal inefficient management of assets.

The Quick Ratio/ Acid Test Ratio
Liquid Assets / Current Liabilities
What is the ideal amount for the Liquid Ratio?
It’s quite similar to the current ratio amount. However, an extremely high Quick Ratio may signal that the company is holding onto too much cash rather than investing it for growth.

The most conservative liquidity ratio: Cash Ratio
Cash / Current Liabilities
What is the ideal amount for the Cash Ratio?
An ideal ratio varies between industries however a range 0.5 - 1 is considered healthy. Anything below 0.5 may be considered risky because the entity has twice as much short-term debt. This ratio is typically used when it's in the process of being acquired by another company.

What are profitability ratios?

The measure of how efficiently a business generates profit from four different things: revenue, assets, equity(the value of an asset after all debts are paid) and capital employed(the amount of money spent by a company to generate profit).

>>> Capital Employed = Total Assets - Current Liabilities or Equity + Noncurrent Liabilities

Margin Ratios:
The measure of how well a business converts revenue into profit. (located in the income statement)
Profit Margin = Type of Profit / Revenue

Net Profit Margin = (Net Profit / Revenue) x 100
Indicates the bottom line profit a business is able to retain for each dollar of revenue earned. It provides a holistic view of a company's profitability because it considers all financial aspects including operating expenses, cost of sales and non operating expenses (e.g. interest, tax).
What is the ideal percentage?
There isn't a single percentage, as the benchmarks on what is considered an ideal NPM varies from industry to industry. Generally, an NPM of 10% is good, and NPM of 20%+ is great and an NPM below 5% is considered undesirable.

Operating Profit Margin = (Operating Profit / Revenue) x 100
Operating profit margin narrows the NPM scope as it emphasizes the operational efficiency of a business by looking at cost of sales, research, development, marketing and admin costs while ignoring interest. It may be helpful for comparing business in different tax jurisdictions.
What is the ideal percentage?
There isn't a single percentage, as the benchmarks on what is considered an ideal OPM varies from industry to industry. Generally, an NPM of 10% is good, and NPM of 20%+ is great and an OPM below 5% is considered undesirable.

Gross Profit Margin = (Gross Profit / Revenue) x 100
Gross Profit Margin is more narrow than OPM as it only considers the direct cost of sales. It helps evaluate a production efficiency against competitors.
What is the ideal percentage?
There isn't a single percentage, as the benchmarks on what is considered an ideal GPM varies from industry to industry; this is extremely evident for GPM ratios. The industry average in retail is about 50% while in sectors like tech or finance the GPM is generally within 80% - 90%

Return Ratios:
Calculates how much Net Profit a business will be able to generate relative to assets or equity or capital employed. (Can be found in the balance sheet. However when doing so make sure to use the average number(opening and closing) resulting from the balance sheet.)

Return on Assets: (Net Profit / Total Assets) x 100
Measures how efficiently a business uses its assets to generate a profit.
What is the ideal percentage?
There isn't a single percentage, as the benchmarks on what is considered an ideal ROA varies from industry to industry. However, a ratio of 5% or higher is generally considered good and a ratio over 20% is considered excellent. Overall, the higher the percentage the better as it indicates effective utilization of assets.

Return on Equity: (Net Profit / Total Equity) x 100

Return on Equity is a direct indicator of how profitable a company is for its investors by showing how efficiently a company is using the money invested by its shareholders to generate earnings.
What is the ideal percentage?
There isn't a single percentage, as the benchmarks on what is considered an ideal ROE varies from industry to industry. A good ROE is usually in between 15% to 20%, with anything above 20% being considered excellent as owners want to maximize their return on the money they invested in. However, depending on other factors a high ROE could display that the business is highly leveraged, increasing financial risk.

Return on Capital Employed: (Operating Profit (also known as EBIT) / Capital Employed) x 100
>>>EBIT stands for earnings before interest and tax.
Measures a company's profitability in terms of all its capital.
What is the ideal percentage?
There isn't a single percentage, as the benchmarks on what is considered an ideal ROCE varies from industry to industry. Overall, the higher the better.

What are efficiency ratios?
The measure of how effective a business is at selling inventory to customers, how quickly it can collect cash back from them (accounts receivable) and how reliably it pays off its creditors.

Turnover Ratios:
The measure of how quickly a business conducts its operations. They compare one line from the Income Statement against a related line in the balance sheet.

Inventory Turnover = Cost of Goods Sold / Inventory
Display how many times a business has sold and restocked its inventory during a set period.
What is the ideal amount?
There isn't a single percentage, as the benchmarks on what is considered an ideal Inventory Turnover Ratio varies from industry to industry. Companies in the retail industry would aim for an Inventory Turnover ratio between 5-10 as it indicates the restocking of inventory every 1 -2 months. While perishable goods industries like grocery stores would have significantly higher turnover ratio.

Receivables Turnover = Revenue / Accounts Receivable
The measure of how effectively a company collects payment from its customers.
What is the ideal amount?
There isn't a single amount, as the benchmarks on what is considered an ideal Receivables Turnover Ratio varies from industry to industry. However, a high ratio suggests effective collection, while a low ratio indicates potential issues with collection.

Asset Turnover = Revenue / Total Assets
Showcases how efficiently a business creates revenue using what it owns.
What is the ideal amount?
There isn't a single amount, as the benchmarks on what is considered an ideal Asset Turnover Ratio varies from industry to industry. A high ratio is considered ideal as it effectively uses its assets to generate earnings minimizing inefficient investment and asset management into the company.

Payables Turnover = Cost of Goods Sold / Accounts Payable
Measures how quickly a company pays off its accounts payable.

What is the ideal amount?
There isn't a single amount, as the benchmarks on what is considered an ideal Payables Turnover Ratio varies from industry to industry. An accounts payable turnover ratio between 6 and 10 is a general rule of thumb across most industries. A high turnover means that the company can pay off supplies at a faster rate and manage cash flow effectively. An extremely high ratio could mean that the business is not reinvesting in itself.

Cash Conversion Cycle:
Indicates the average number of days a business needs to convert its investments and inventory into cash.
Cash Conversion Cycle = Days Sales of Inventory + Days Sales Outstanding + Days Payable Outstanding

Days Sales of Inventory = (Inventory / Cost of Goods Sold) x 365
Calculates the average number of days it takes a business to convert its inventory into sales. This ratio is also known as the inventory turnover period.
What is the ideal amount?
There isn't a single amount, as the benchmarks on what is considered an ideal DSI Ratio varies from industry to industry. However, a good DSI is considered to be between 30-60 days. A lower DSI is generally preferred but an extremely low one can indicate that the company does not enough inventory to meet demand.

Days Sales Outstanding = (Accounts Receivable / Revenue) x 365
Calculates the average number of days in a receivables collection period. Helps assess cash flow management.
What is the ideal amount?
There isn't a single amount, as the benchmarks on what is considered an ideal DSO Ratio varies from industry to industry. However, a low DSO ratio is an ideal as it indicates efficient cash management and quicker access to cash for reinvestment.

Days payable Outstanding = (Accounts Payable / Cost of Goods Sold) x 365
Calculates the average number of days in a payables turnover period.
What is the ideal amount?
There isn't a single amount, as the benchmarks on what is considered an ideal DPO Ratio varies from industry to industry. A high DPO can be desirable based on the company's goals, industry and other factors as it symbolizes the use of available cash for short-term investments as well as to increase their capital and free cash flow by delaying payments. Or it could represent a cash short-fall and the company's inability to pay.

Leverage Ratios:
Leverage is when you take a higher risk by taking on debt in order to maximize your return or reward. They are two types of leverage ratios; one located in the balance sheet and the other in the income statement.

Balance Sheet Ratios:
The measure of a company's debt level relative to its equity and assets, indicating how much it relies on borrowing to carry out its operations.
Balance Sheet Ratios = Total Liabilities / Total Assets or Equity

Debt to Assets = Total Liabilities / Total Assets
Calculates how much the business's assets have been financed using debt by considering both long and short-term debt obligations.
What is the ideal amount?
There isn't a single amount, as the benchmarks on what is considered an ideal DTA Ratio varies on company

size, sector and industry. The more secure a company is, the easier it is to obtain loans leading to higher ratio vs. a startup with multiple private investors with a lower ratio. In general, the ideal ratio is around 0.3 - 0.6.

Debt to Equity = Total Liabilities / Equity
Compares a company's total liabilities with its shareholder equity. It is used to indicate the extent of a business's reliance on debt.
What is the ideal amount?
There isn't a single amount, as the benchmarks on what is considered an ideal D/E Ratio varies on company size, sector and industry. A D/E ratio below 1 is considered safe, while 2+ may be viewed as risky.

Income Statement Ratios:
These ratios determine a business's ability to meet its financial obligations.
Income Statement Ratio = Type of Profit / Type of Interest

Interest Coverage = Operating Profit / Interest Expense
Since operating profit is calculated before interest the equation depicts whether a business has earned enough profit to cover the interest of its debt obligations.
What is the ideal amount?
There isn't a single amount, as the benchmarks on what is considered an ideal D/E Ratio varies on company size, sector and industry. However, a relatively low ratio means that a company can reliably cover its interest payments.

Total Debt Service = Interest + Principle

Debt Service Coverage Ratio = EBITDA / Total Debt Service
Is used to evaluate whether a firm can use its available cash flow to pay its current obligations.
What is the ideal amount?
There isn't a single amount, as the benchmarks on what is considered an ideal DSCR varies on company size, sector and industry. A DSCR above 1.25 is often considered strong with 2.00 being excellent. A ratio below 1.00 may signal that the business is facing financial issues.

Price Ratios:
Investors use price ratios in financial analysis to evaluate the share price of a business and determine if it is a good investment.

Earning Ratios:

Earnings Per Share = Net Profit or (Net Profit - Preferred Dividends) / Common Shares Outstanding
Represent the amount of a business's profit that's allocated to each share of a common stock.
What is the ideal amount?
There isn't a single amount, as the benchmarks on what is considered an ideal EPS varies on company size, sector and industry. It also depends on factors like the recent performance of the company, its competitors and analyst expectations.

Price to Earnings Ratio = Share Price / Earnings Per Share
Displays how much the market is prepared to pay for each dollar of earnings.
What is the ideal amount?

There isn't a single amount, as the benchmarks on what is considered an ideal P/E varies on company size, sector and industry. To get a rough idea of whether a P/E ratio is high or low, compare it to competitor P/E ratios. Generally, a lower P/E ratio is considered better because investors pay less for every dollar or earnings.

Price/Earnings to Growth Ratio = Price to Earnings Ratio / Earnings Per Share Growth
Is used to determine a stock's value while factoring in expected earnings growth.
What is the ideal amount?
There isn't a single amount, as the benchmarks on what is considered an ideal PEG varies on company size, sector and industry. The lower the PEG, the more the stock may be undervalued as analysts predict its growth to be limited. A high PEG may be a sign of a company to have a high growth rate and P/E ratio.

Dividend Ratios:

Dividend Per Share = Dividends Paid or (Dividends Paid - Special Dividends)/ Common Shares Outstanding
The sum of declared dividends issued by a business per every ordinary outstanding share. Investors gain a rough idea of the potential income they may gain.
What is the ideal amount?
There isn't a single amount, as the benchmarks on what is considered an ideal DPS varies on company size, sector and industry. A good DPS generally ranges between 2% to 6% of the stock price.

Dividend Yield Ratio = Dividends Per Share / Share Price
Depicts a business's yearly dividends relative to its shares' market price.
What is the ideal amount?
There isn't a single amount, as the benchmarks on what is considered an ideal Dividend Yield Ratio varies on company size, sector and industry. A high dividend may not always be great, as a company may be better benefited by reinvesting itself to expand and reward shareholders with higher capital gains. Generally les than 4% is considered safe as higher percentages increase risk.

Dividend Payout Ratio = Dividends Paid / Net Profit
It is the ratio of the total amount of dividends paid out to shareholders in accordance to the net income of the company.
What is the ideal amount?
There isn't a single amount, as the benchmarks on what is considered an ideal Dividend Payout Ratio varies on company size, sector and industry. A high dividend may not always be great, as a company may be better benefited by reinvesting itself to expand. It may use a high dividend to mask a bad decision.

Risk Adjusted Return Ratios
A calculation of the earnings or potential profit from an investment that factors in the amount of risk that must be considered to obtain it.

Risk:
The probability of losing money over a certain period.

Risk-Free Rate:
Is the return an investor should be able to expect assuming there is no risk associated with the investment.

Downside RIsk:

The potential loss that the investor may experience.

Standard deviation

$$\text{Standard Deviation} = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \overline{x})^2}{n-1}}$$

where:

$x_i = \text{Value of the } i^{th} \text{ point in the data set}$

$\overline{x} = \text{The mean value of the data set}$

$n = \text{The number of data points in the data set}$

Measures the dispersion of a dataset compared to its mean. It is calculated as the square root of the deviation (incongruity from the mean).
How to do it?

1) FInd mean amongst data points
2) Calculate variance: subtract mean for each data
3) Square the variance from each data point
4) Sum the squared variance values
5) Find average: Divide sum(Step 4) by the no. of data points minus 1. (n -1)
6) Square root Step 5

What is the ideal amount?
There isn't a single amount, as the benchmarks on what is considered an ideal Standard Deviation varies on company size, sector, industry and investor preference. Investors must consider their tolerance for volatility and objectives. For a conservative investor, a low standard deviation risk may be ideal.

Alpha

$$\text{Alpha Formula} = \text{Actual Rate of Return of Portfolio} - \text{Expected Rate of Return on Portfolio}$$

A measure of an investment's performance that indicates ability to generate returns above the investment's benchmark in its asset category.
What is the ideal amount?
There isn't a single amount, as the benchmarks on what is considered an ideal Alpha varies on company size, sector, industry and investor preference. Overall, an alpha greater than zero is considered good.

Beta

$$\text{Beta coefficient}(\beta) = \frac{\text{Covariance}(R_e, R_m)}{\text{Variance}(R_m)}$$

where:

$R_e = \text{the return on an individual stock}$

$R_m = \text{the return on the overall market}$

$\text{Covariance} = \text{how changes in a stock's returns are related to changes in the market's returns}$

$\text{Variance} = \text{how far the market's data points spread out from their average value}$

An volatility indicator that compares a stock to the broader market.
What is the ideal amount?
There isn't a single amount, as the benchmarks on what is considered an ideal Beta varies on company size, sector, industry and investor preference (a conservative vs. aggressive approach).
A beta <1 means that the stock is less volatile in the market.
A beta >1 means that the stock is more volatile in the market.
A beta =1 means that the stock corresponds with the market.
A beta <0 means that the stock is inversely correlated with the market.

Sharpe Ratio

$$Sharpe\ Ratio = \frac{R_p - R_f}{\sigma_p}$$

where:

$R_p = \text{return of portfolio}$

$R_f = \text{risk-free rate}$

$\sigma_p = \text{standard deviation of the portfolio's excess return}$

Measures the excess return compared to the risk-free rate per unit of risk. Most widely used risk-adjusted ratio.
What is the ideal amount?
There isn't a single amount, as the benchmarks on what is considered an ideal Sharpe Ratio varies on company size, sector, industry and investor preference. However, Sharpe Ratios above 1 is generally considered good and a sharpe ratio of 1.5+ is considered to be excellent.

Sortino Ratio

$$\text{Sortino Ratio} = \frac{R_p - r_f}{\sigma_d}$$

where:

$R_p = \text{Actual or expected portfolio return}$

$r_f = \text{Risk-free rate}$

$\sigma_d = \text{Standard deviation of the downside}$

A variation of the sharpe ratio that only focuses on the volatility of negative returns.

What is the ideal amount?
There isn't a single amount, as the benchmarks on what is considered an ideal Sortino Ratio varies on company size, sector, industry and investor preference. Generally, a sortino ratio above 1 is considered good, while 2+ signifies excellent performance.

Treynor Ratio

$$\text{Treynor Ratio} = \frac{r_p - r_f}{\beta_p}$$

where:

$r_p = \text{Portfolio return}$

$r_f = \text{Risk-free rate}$

$\beta_p = \text{Beta of the portfolio}$

Calculates the return earned in excess of the risk-free return. It is similar to the Sharpe Ratio but doesn't utilize standard deviation.
There isn't a single amount, as the benchmarks on what is considered an ideal Treynor Ratio varies on company size, sector, industry and investor preference. A good ratio reflects a higher return when compared to the portfolio's exposure to market risk.

Jenson Ratio

Alpha = R(i) - (R(f) + B x (R(m) - R(f)))

Where:

- R(i) = the realized return of the portfolio or investment
- R(m) = the realized return of the appropriate market index
- R(f) = the risk-free rate of return for the time
- B = the beta of the investment portfolio related to the chosen market index

Measures how much an investment returned above or below its expected return defined by the capital asset pricing model(CAPM).
What is the ideal amount?
A positive value means that the expected return has been exceeded, while a negative means the company earned less than the expected return when adjusted for risks. Zero means that the alpha is neutral and corresponds with the market or benchmark.

Research: SHAP plot types

Types of SHAP Plots:
Summary plots

- Ranks features based on their effect for model predictions (x - axis representing SHAP values - a measure depicting quantitative influence)
- The y -axis is feature distribution, with the colour red representing high importance and blue with low low

importance
- Each dot represents a specific data point's SHAP value
- The graph looks like a beeswarm

Bar plots
- The SHAP bar plot presents each feature's absolute SHAP values as bars in chart form
- The taller the bar, the more significant the importance to the model's decisions
- Positive values represent positive influence, and negative values represent negative influence

Force plots
- The force plot offers an in-depth perspective of SHAP values for singular occurrences
- It shows the base value (expected model output) and demonstrates how each feature's influence pushes the model's predictions
- Emphasizes positive and negative impacts better than the bar plot because by illustration of how each feature contributes to predictions through variance with changing values and how each decision evolves when features are added or removed.

Waterfall plots
- Useful visualization tool that displays the additive contributions of features to a model's prediction for a specific instance
- A type of force plot that begins with a baseline value and shows each figure's contribution to the overall prediction

Heatmaps
- Creates a plot with instances on the x-axis and model inputs on the y-axis
- SHAP values are encoded on a colour scale

What will be the best implementation for my project?
The "Waterfall" plot is simple, sleek and visually easy to comprehend, with its format not only representing the weight of each decision but also the final choice made by the algorithm, making it the perfect decision to use in my project

Data

Making a glossary

Conclusion
Retail trading apps
Backtesting software
Educational platforms

Citations

Acknowledgements

Attachments
Financial Risk

Business Quality

Market Uncertainty

Positioning

Market momentum

Cognitive Concept	Ratio/Metric	The 'Because' logic
Financial Risk	Debt-to-Equity	It is used to indicate the extent of a business's reliance on debt. a.k.a how much they have vs. borrowed
Business Quality	Net Profit Margin	Indicates the bottom-line profit a business is able to retain for each dollar of revenue earned.
Market Uncertainty	Bollinger Band Width	3 lines that encompass 95% of the stock and indicate volatility (the smaller the width, the more stable
Positioning	%B (price location)	Where the stock is within the Bollinger band. The higher it is, the more overbought it is. The lower it is, the more oversold it is.
Market Momentum	RSI	It provides technical traders with information about whether market conditions are bullish (on the rise) or bearish (falling). An asset is considered overbought when above 70 and oversold at 30.

Research refined

1. Aspects/Issues explored by this question

To simulate a professional portfolio, I decided to organize my selected stocks into three risk-stratified categories, drawing inspiration from my role as the Lead Quantitative Analyst in

the Wharton Junior Investment Program, where I first introduced the initial model when working alongside my team. Together, we refined it and devised a “Triathlon” analogy that perfectly describes the level of volatility(sensitivity to the market) in each category:

1. The “Swim”(Conservative): Comprised of ETFs like AGG, BSV, SCHD. These assets are centred on broad diversification, price stability, and long-term longevity, acting as an anchor of the portfolio.
2. The “Run”(Moderate): Consisting of a mix of ETFs and stocks like VOO, MSFT, CEG, WCN, and AMT. These are industry leaders known for consistent growth and established market share within their respective sectors.
3. The “Bike”(Aggressive): Made up of QQQ, ICLN, PAVE, NVDA, AVAV. These tech stocks and ETFs contribute to short-term exponential portfolio growth.

NOTE*** While the “Triathlon” categorization was developed during my extracurricular program, all other elements in this project were developed independently for this research.

—----------

Financial ratios and metrics

After analyzing over 35 financial ratios/metrics within the categories of price, profitability, efficiency, liquidity, leverage, and risk-adjusted return, I selected 5 main metrics that would be integrated into my xAI explanations.

Financial Risk

Debt to Equity = Total Liabilities / Equity

Compares a company’s total liabilities with its shareholder equity. It is used to indicate the extent of a business’s reliance on debt.

What is the ideal amount?

There isn’t a single amount, as the benchmarks on what is considered an ideal D/E Ratio varies on company size, sector and industry. A D/E ratio below 1 is considered safe, while 2+ may be viewed as risky.

Business Quality

Net Profit Margin = (Net Profit / Revenue) x 100

Indicates the bottom-line profit a business is able to retain for each dollar of revenue earned. It provides a holistic view of a company’s profitability because it considers all financial aspects, including operating expenses, cost of sales and non-operating expenses (e.g. interest, tax).

What is the ideal percentage?

There isn’t a single percentage, as the benchmarks on what is considered an ideal NPM vary from industry to industry. Generally, an NPM of 10% is good, and an NPM of 20%+ is great, and an NPM below 5% is considered undesirable.

Market Uncertainty

The distance between the mean and the upper/lower Bollinger band(set at 2 standard deviations) that encompass 95% of the stock. The greater the bandwidth, the more volatile the stock/ETF is.

What is the ideal amount?

There isn't an ideal amount; however, compressed Bollinger bands are preferred to signify market stability.

Positioning
The location of the stock is in alignment with the Bollinger band's brackets. If it is located above the median, the stock is considered overbought and overvalued and below is oversold and undervalued.

Market Momentum
The Relative Strength Index is a technical indicator in momentum trading that is a tool used to measure the speed and magnitude of a recent price change in a stock.
Refer to the RSI Confluence Section

—----------

To engineer my product, I used the VSCode platform and the principles of Object-Oriented Programming(OOP) to separate the algorithm's processing, interpretation and interface.

i) Language: Python
Known for being the domanint language in the field of algorithmic trading, Python3 offers extensive libraries like pandas, numpy and scikit-learn for data manipulation, analysis, strategy backtesting and machine learning. Facilitating multiple API's that fetch dynamic market-place data to carry out orders. Thus, making it the ideal language for prototyping, research, and running such systems.

After preforming visual scripting on the Unity platform, I only got introduced to language scripting in september to javascript and java for Computing Science 20. In October, when I decided to participate in CYSF, I took 2 online courses from Kaggle ("Python" and "Intro to Machine Learning"), to help me understand fundamental syntax and construction of machine learning.

ii) APIs
A set of rules and procedures enabling the creation of systems that access data of an operating system or application.
To learn data manipulation and plotting from pandas, numpy and matplotlib, I referred to the youtube videos made by "Bro Code." For yfinance and shap I used a variety of websites and youtube videos. However, there was a lacking of sources to convert data into "wrapper" to fit shap, so I utilized an LLM to aid the niche conversion.

API	Main Role	Purpose for project
yfinance	Data Sourcing	Sourced historical data from beginning November 1st 2019, to ensure the Bollinger Bands Mean Reversion strategy was tested across multiple market conditions.
pandas	Data Cleaning	Used for data cleaning of the input received by the finance API. It handled missing data to prevent the code form crashing, managed the 2D dataframes for risk-stratification.
numpy	Data manipulation	Enabled quick math conversions to reduce the hassle of redundant loops and handled the math for internal SHAP logic.
matplotlib	Human Interface	Created the Interactive Scanner, enabling a cleaner output to display a clear signal associated with a specified section
shap	xAI Engine	Provided the Axiomatic Attribtuion. shap calculated the mathematical weight of each factor (Financial Risk, Market Momentum e.t.c) and when combined with plain english, produced an eli5 response.
tabulate	Glossary Formatting	Enabled a dynamic glossary to provide users with the purpose of each element used to back up the algorithm's choice.

——----------

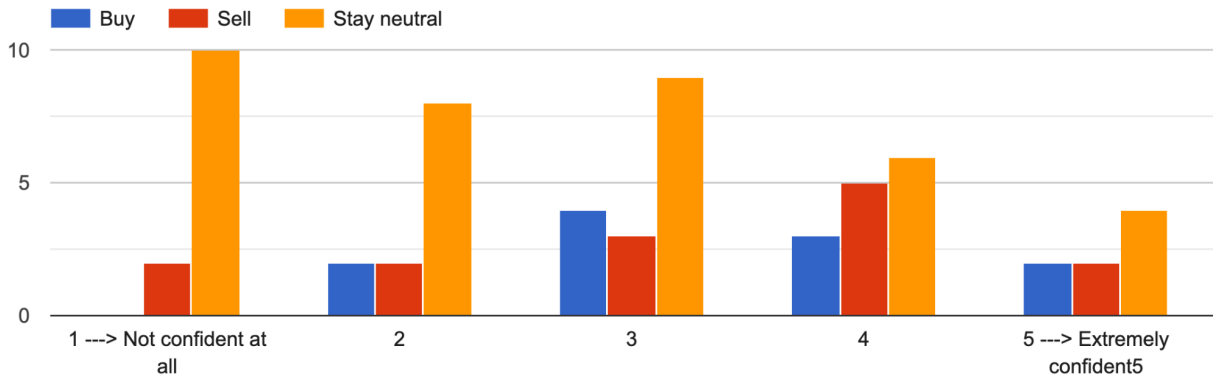
Question	order	SHAP Group	Pos diff	ELI5 group	Pos diff
Prior Experience: On a scale of 1-10, how familiar are you with stock market trading or algorithmic signals?	1	3.689655172 Data distribution: A mix of Gaussian overpowered by a downward trend. The majority of 7 for a confidence of one is much more extremely unconfident than ELI5, yet still more confident as a whole	+.3	3.379310345 Data distribution: Downwards(Probability mass function/Poisson distribution) with a majority of 7 for a confidence of 2	
How confident are you that these are a good trade	2	5.275862069 Guasian with a a majority	+.9	4.448275862 Data distribution: Roughly normal	

based solely on this visual from the following "Bollinger Band" algorithm? (look at the 2nd box in the top left.)		of 5 form 8 peeps More people were trusting than ELI5		2 people said 10 6 peeps for 5 No one for 8 or 9 Leaning towards distrust	
Mental Model Accuracy: If the stock price hit the upper Bollinger Band and the RSI is 75, what would you think the algorithm will signal? From a scale of 1-5 how confident are you in this choice?	3 Ans = Sell	Distribution wa more poission, most said neutral but the second mosts popular answer is sell like ELI5 The group with most correct answers was 4 withonly one more vote towards neutural		Data Distribution: Gaussian but broken by 1 (not confident at all) The majority said stay neutral, but in second was sell pretty close (-16 with 39 votes across neutral) Can't say because the data has more results than the number of people participating	
Now that you see the xAI's explanation, how confident are you in this signal? (1-10)	4	5.068965517 Manily gaussian with 12 and only one more person who is >5 no one chose 9	+.4	4.724137931 Mainly Gaussian with 12, but 4 and 8 are equal with 3 peeps, no one said 9 or 10	
Clarity: From a scale of (1-10) how easy was it to understand why the AI made this decision?	5	4.206896552 Data Type: No trend 1st place: 6 peeps said 1 and 5 with a close second of 8 from 4 peeps no one for 9 or 10		4.517241379 Data type: relatively straight The majority: 5 with 6 peeps The highest: 9 with 3 peeps The second highest: 2 with 5 peeps A common number of dibursion was roughly 3 No one said 7 or 10	+.3
Mental Model Accuracy: If the stock price hit the lower Bollinger Band and the RSI is 72, what would you think the algorithm will signal? From a scale of 1-5 how confident are you in this choice?	6 Ans = Neutural	The answer is neutral and correct with a string majority but this raises question on always being the most predominant answer. A shocking thing is that neutral is the weakest sectionw here there is a confidence of 4.		They got it right with a majority staying neutral, but raises questions from the results of whether staying neutral is the right choice. Less confidence with this answer. With the majority of neturals in 1 not confident at all. Data distribution was downwards	

Now that you have seen another xAI explanation, how confident are you in this signal? (1-10)	7	4.689655172 Guassian distribution. 11 people said 5 2 sais 10 and 2 said 1 ruth no one for 8 or 9, most leaning towards less confideance Confidene decreased by 0.4		4.655172414 Gaussian distribution 8 peeps said 5 6 peeps said 6, and 5 peeps said 5 8 and 3 match with 3 No one said 9 or 10 Confidence decreased -0.1	
Please explain why your answer improved, unimproved or stayed the same from your original answer. (250 words max)		Despite a confidence decrease most individuals couldn't explain why they felt change.			
Mental Model Accuracy: If the "Financial Risk" feature increased significantly, but the stock price stayed the same (near the highest band as a 55 RSI) , what would you expect the AI to recommend? How confident are you with this answer?	8 Ans = Neutural but sell acceptable	Poisson distribution Majority where correct by neuttual but only a clea winner with the unconfident as we get more confident the increase in sell comes with is an acceptable answer		Again least confident majority said 1. Neutral is correct but raises whether people just show that for choosing, but there is a slightly bigger discrepancy for sell in the least confident seciton. In other sections results were really close. In 3 tie happened between sell and huetural though, however, sell is acceptable.	
The xAI suggests to 'BUY' even though market conditions appear to be dropping. Does the provided explanation make you more or less likely to follow the signal? (1 -10)	9	5.068965517 10 people said 5 Followed gaussian distribution and no on one said 6 the second most predominant answer is 7 with 5 peeps		5.103448276 13 people said neutral, and there was an even distribution of people across both sides, with one more person for yes, they would still trust the AI signal.	
If this AI were wrong in a real-world scenario, how would that	10	It would have a great negative impact on most who have an aversion towards ai automate tools, the main reason being sensitivity towards money.		It would have a great negative impact on most who have an aversion towards ai automate tools	

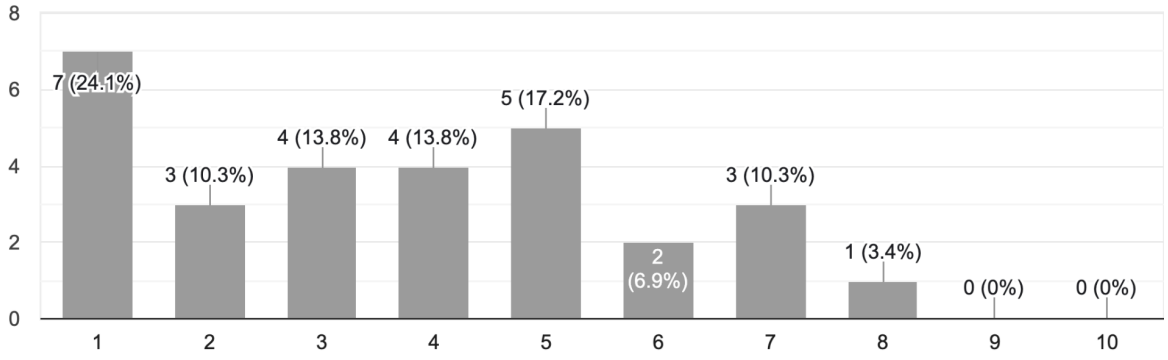
impact your trust in automated financial tools?					

Mental Model Accuracy: If the stock price hit the upper Bollinger Band and the RSI is 75, what would you think the algorithm will signal? From a scale of 1-5 how confident are you in this choice?



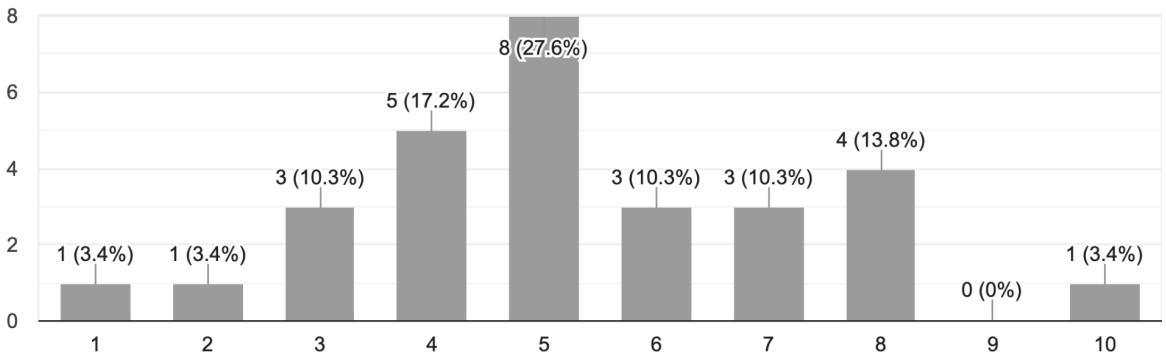
Prior Experience: On a scale of 1-10, how familiar are you with stock market trading or algorthimic signals?

29 responses

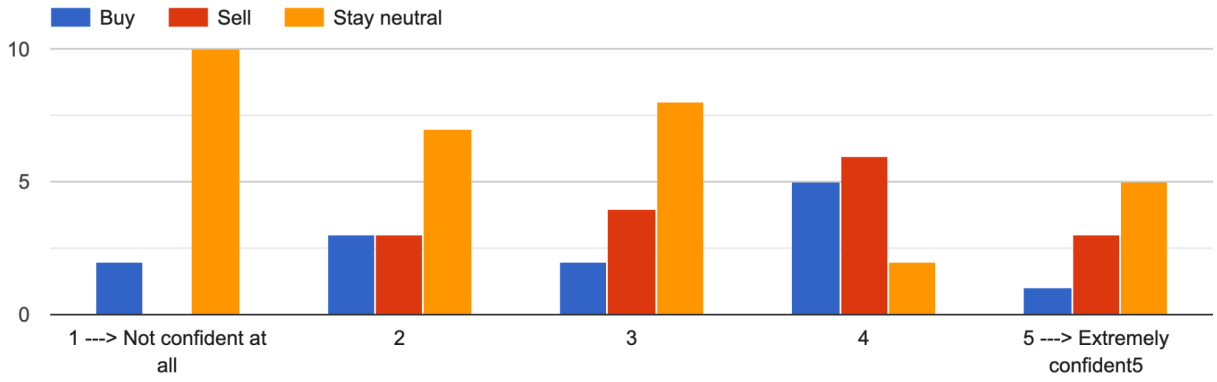


How confident are you that these are a good trade based solely one this visual from the following "Bollinger Band" algorithm? (look at the 2nd box in the top left.)

29 responses

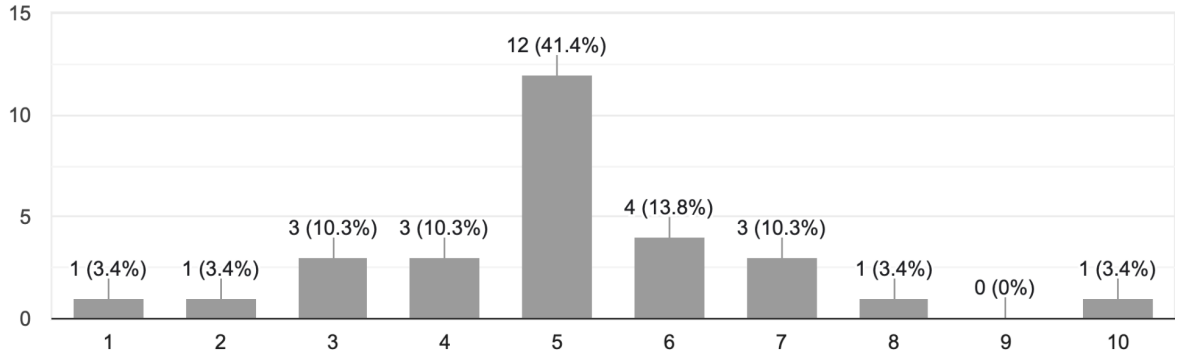


Mental Model Accuracy: If the stock price hit the lower Bollinger Band and the RSI is 72, what would you think the algorithm will signal? From a scale of 1-5 how confident are you in this choice?



Now that you see the xAI's explanation, how confident are you in this signal? (1-10)

29 responses



Clarity: From a scale of (1-10) how easy was it to understand why the AI made this decision?

29 responses