Portfolio Python

BY: STEVE VOGEL

Introduction

- Select Stocks (2-4) @ current risk with 3 yr predicted Close
- Input Variable: Historical Close
- Target Variable: Predicted Close (3 yrs) (Reflecting Investor Retire date)
- Worked/Didn't work: Will Review by model
- Monte Carlo chosen as accurate historical predictor
- FB Prophet chosen as a well functioning predictor (separate
  and seasonality from data.)

Cupcake, Birthday Cake and Wedding Cake

	Analyze Portfolio	Optimization of Portfolio	Portfolio Selection
	Cupcake	Birthday Cake	Wedding Cake
Models	Pandas	FB Prophet	Regression Anal, Lin Regression,
		Machine Learning sklearn	Neural Network- TensorFlow, Keras
		Random Forest	Dashboard
		Monte Carlo	
Data Source	CSV, ALPACA	ALPACA Yfinance,	API
Variables	Closing Price	Closing Price, Returns	Closing Price, Returns
	NASD, SPY		
	DJIA	News Stock, Rumors	
Code Structure	Stock,Pandas, PIP Install	Stock Class	
	Functions	LSTM Stk Pred	StreamSt
		Portfolio Class	StatsModel
		Functions	Time Series
Visualization	pathlib, plotlib	plot_plotly,	
Investments	ADP, BR, CDK, AAPL, GOOG, NFLIX, NVDA	AAPL, AMZN,GOOG, NVDA, NTFX, FB, PFE, (BIIB -Bio place holder) 8 Stocks	BTC, ETH, SOL, ADA (DOGE, AVAX (narrow

FB Prophet (Model 1) Code Slide 1

In [1]:
```
import streamlit as st
from path import Path
from datetime import date
import pandas as pd
```

In [2]:
```
import yfinance as yf
from prophet import Prophet
from fbprophet.plot import plot_plotly
from plotly import graph_objs as go
START = "2016-01-01"
TODAY = date.today().strftime("%Y-%m-%d")
st.title("Anthon Stock Predictor")
stocks = ("AAPL","AMZN","GOOG", "NTFX", "NTFX", "ORCL", "TSLA", "FB", "PFE", "BIIB")
selected_stocks = st.selectbox("Select dataset for prediction", stocks)
n_years = st.slider("Years of Prediction:", 1,4)
period = n_years * 365
#(venv) stock prediction code
#(venv) stock prediction streamlit run main.py
def load_data(ticker):
    data = yf.download(ticker, START, TODAY)
    data.reset_index(inplace=True)
    return data

##Set stock to analyze
selected_stock = "AMZN"

data_load_state = st.text("Load data...")
data = load_data(selected_stock)
data_load_state.text("Loading data...done!")

# Forecasting
df_train = data[['Date', 'Close']]
df_train = df_train.rename(columns= {"Date":"ds","Close":"y"})

m=Prophet()
m.fit(df_train)
future = m.make_future_dataframe (periods=period)
forecast = m.predict(future)

print("AMAZON:")

st.subheader ('Forecast data')
st.write(forecast.tail())
st.write('forecast data')
#Fig1 = plot_plotly (m.forecast)
st.write('forecast components')
Fig2 = m.plot_components(forecast)
st.write(Fig2)
```

FB Prophet Code 2

In [3]:

```python
##Set stock to analyze
selected_stock = "GOOG"

data_load_state = st.text("Load data...")
data = load_data(selected_stock)
data_load_state.text("Loading data...done!")

# Forecasting
df_train = data[['Date', 'Close']]
df_train = df_train.rename(columns= {"Date":"ds","Close":"y"})

m=Prophet()
m.fit(df_train)
future = m.make_future_dataframe (periods=period)
forecast = m.predict(future)

print("GOOGLE:")

st.subheader ('Forecast data')
st.write(forecast.tail())
st.write('forecast data')
#Fig1 = plot_plotly (m.forecast)
st.write('forecast components')
Fig2 = m.plot_components(forecast)
st.write(Fig2)
```

FB Prophet Code Slide 3

```
##Set stock to analyze
selected_stock = "PFE"

data_load_state = st.text("Load data...")
data = load_data(selected_stock)
data_load_state.text("Loading data...done!")

# Forecasting
df_train = data[['Date', 'Close']]
df_train = df_train.rename(columns= {"Date":"ds","Close":"y"})

m=Prophet()
m.fit(df_train)
future = m.make_future_dataframe (periods=period)
forecast = m.predict(future)

print("PFE:")

st.subheader ('Forecast data')
st.write(forecast.tail())
st.write('forecast data')
#Fig1 = plot_plotly (m.forecast)
st.write('forecast components')
Fig2 = m.plot_components(forecast)
st.write(Fig2)
```

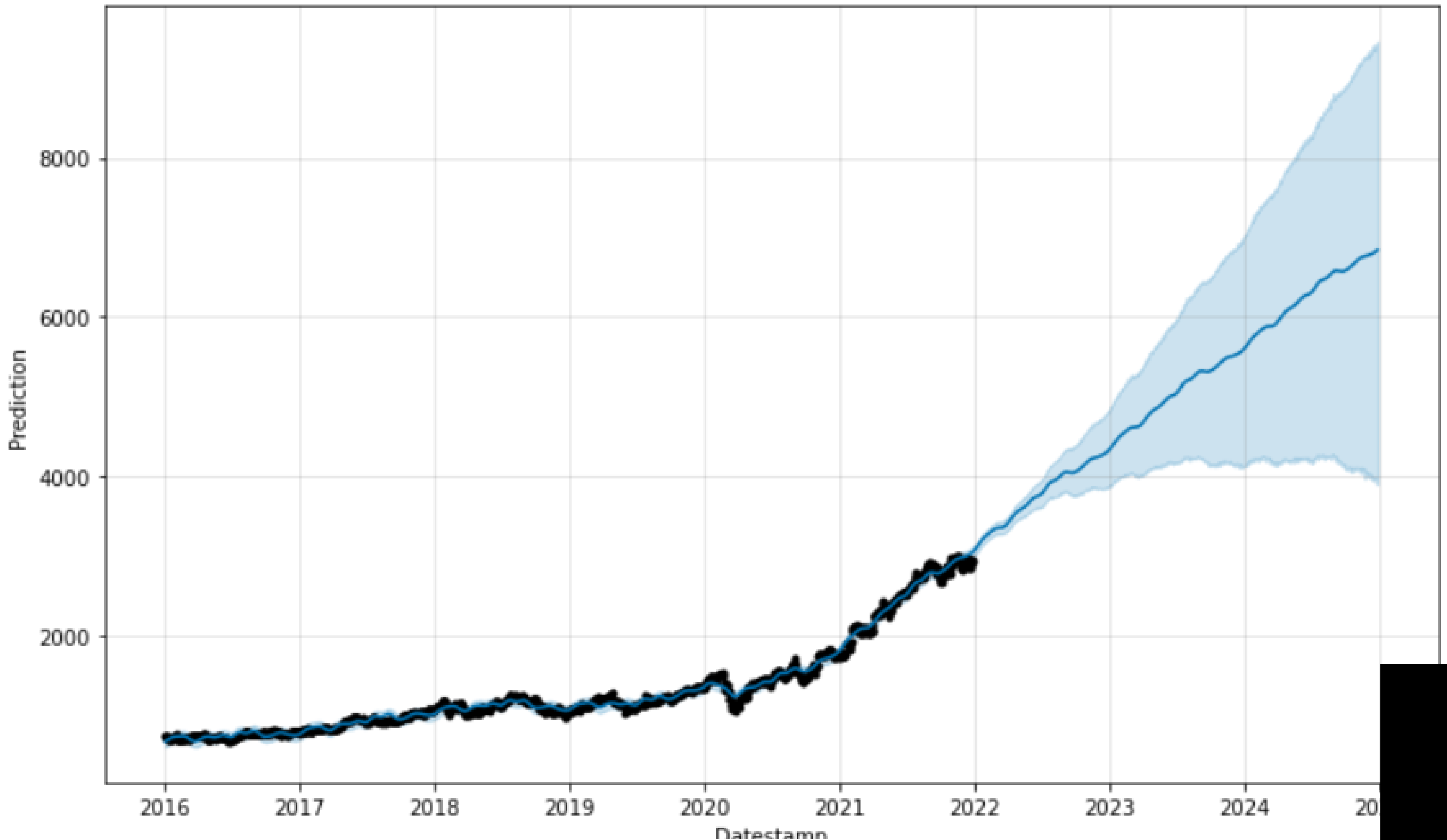
FB Prophet Code 4

In [5]:

```python
##Set stock to analyze
selected_stock = "NFLX"

data_load_state = st.text("Load data...")
data = load_data(selected_stock)
data_load_state.text("Loading data...done!")

# Forecasting
df_train = data[['Date', 'Close']]
df_train = df_train.rename(columns= {"Date":"ds","Close":"y"})

m=Prophet()
m.fit(df_train)
future = m.make_future_dataframe (periods=period)
forecast = m.predict(future)

print("NETFLIX:")

st.subheader ('Forecast data')
st.write(forecast.tail())
st.write('forecast data')
#Fig1 = plot_plotly (m.forecast)
st.write('forecast components')
Fig2 = m.plot_components(forecast)
st.write(Fig2)
```

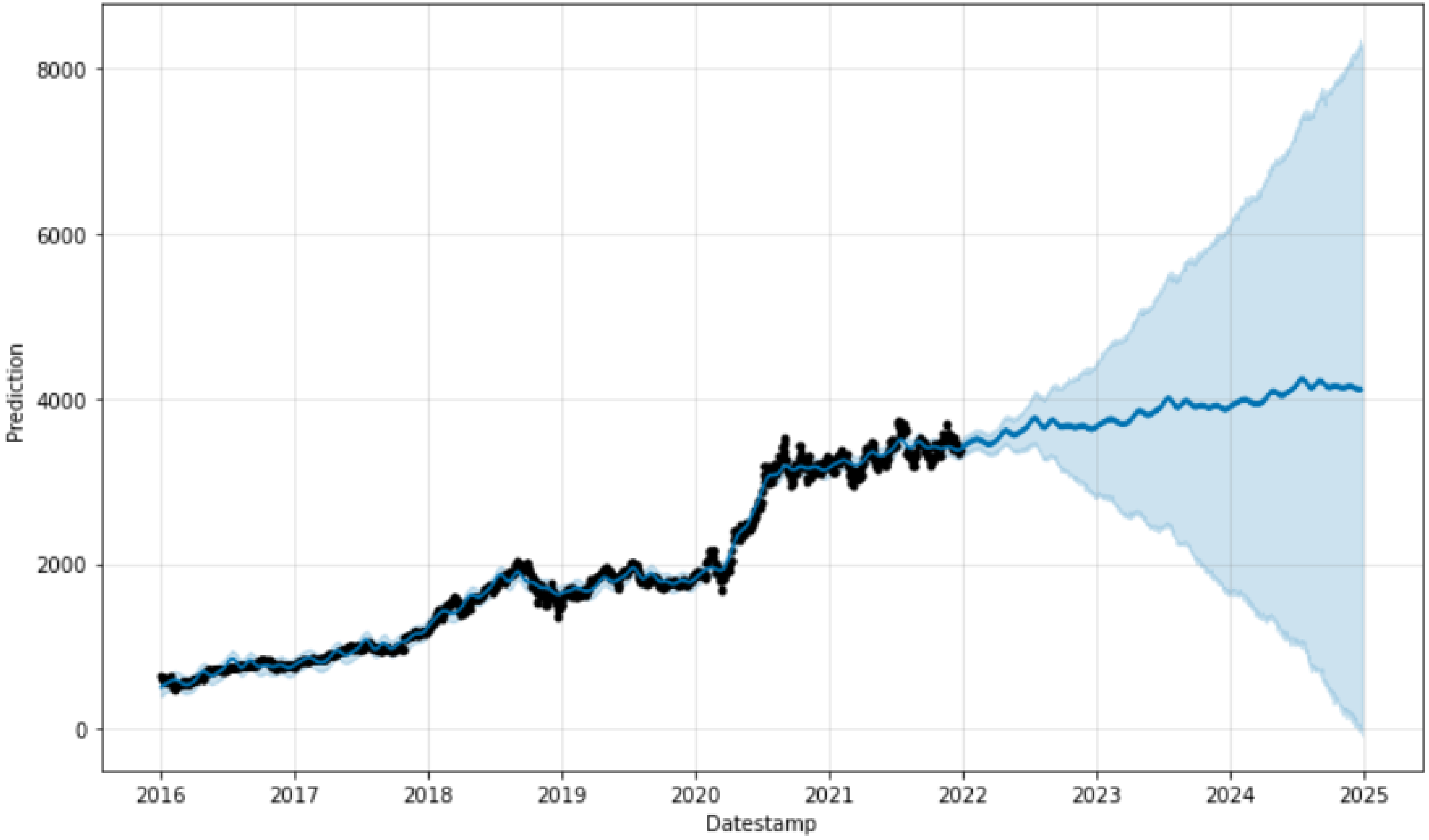
Googol Close Price Trend (USD)

GOOG Daily plot:

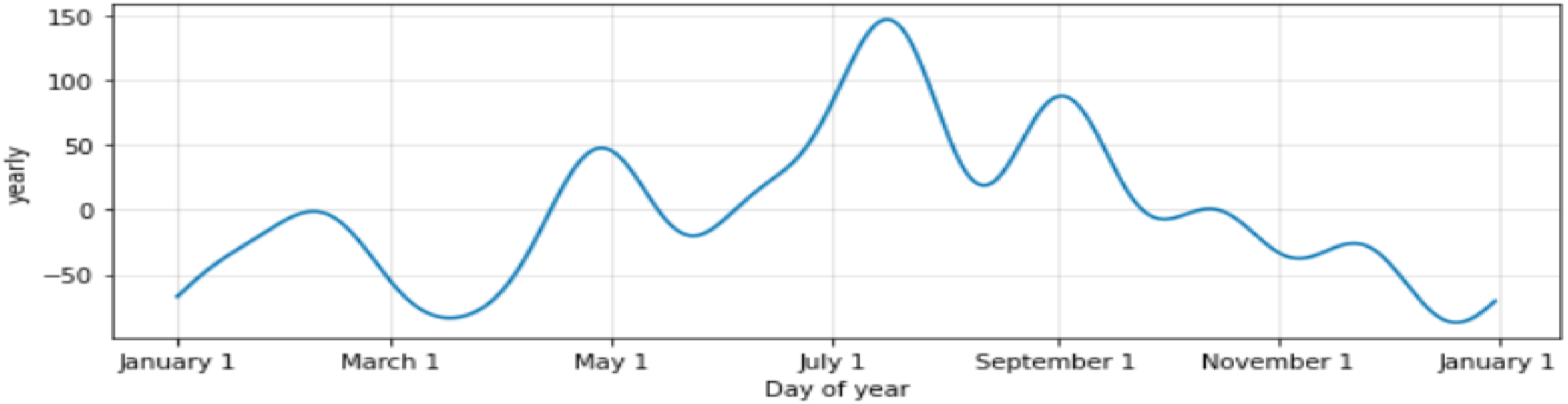


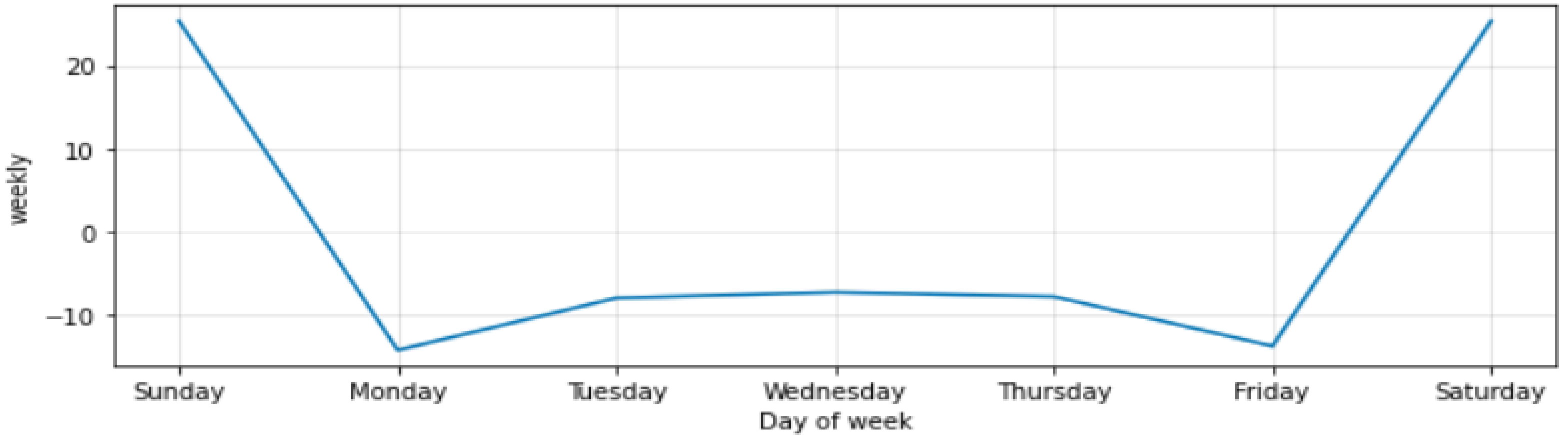
Amazon Close Price Trend (USD)

Daily plot:



YEARLY SEASONALITY (USD) above
WEEKLY VARIABILITY (USD) below





MC Simulation Code Slide1

In [1]:

```
# Initial imports
import os
import requests
import pandas as pd
from dotenv import load_dotenv
import alpaca_trade_api as tradeapi
from MCForecastTools import MCSimulation

%matplotlib inline
```

In [2]:

```
# Load .env environment variables
load_dotenv()
```

Out[2]: True

In [3]:

```
# Set start and end dates of three years back from today.
start_date = pd.Timestamp('2019-01-01', tz='America/New_York').isoformat()
end_date = pd.Timestamp('2021-12-17', tz='America/New_York').isoformat()
```

In [4]:

```
# Set Alpaca API key and secret
alpaca_api_key = os.getenv("ALPACA_API_KEY")
alpaca_secret_key = os.getenv("ALPACA_SECRET_KEY")

# Create the Alpaca API object
api = tradeapi.REST(
    alpaca_api_key,
    alpaca_secret_key,
    api_version = "v2"
)
```

MC Simulation Code 2

5]:

```
# Get 5 years' worth of historical data for AMZN, GOOG
tickers = ["AMZN", "GOOG"]

# Set timeframe to '1D' for Alpaca API
timeframe = "1D"

# Get current closing prices for AMZN, GOOG
df_stock_data_1 = api.get_barset(
    tickers,
    timeframe,
    start=start_date,
    end=end_date,
    limit=756
).df

next_start_date = pd.Timestamp('2017-01-01', tz='America/New_York').isoformat()
next_end_date = pd.Timestamp('2019-01-01', tz='America/New_York').isoformat()

df_stock_data_2 = api.get_barset(
    tickers,
    timeframe,
    start=next_start_date,
    end=next_end_date,
    limit=756
).df

df_stock_data = pd.concat([df_stock_data_2, df_stock_data_1])

# Display sample data
df_stock_data.head()
```

MC Simulation Code 3
Dataframe heads

:[5]:

	AMZN					GOOG				
	open	high	low	close	volume	open	high	low	close	volume
time										
2017-01-03 00:00:00-05:00	757.92	758.7595	747.7000	753.66	2511913	778.81	789.6300	775.8000	786.14	1061256
2017-01-04 00:00:00-05:00	758.24	759.6800	754.2000	757.18	1671835	788.36	791.3400	783.1600	786.87	634357
2017-01-05 00:00:00-05:00	761.55	782.3999	760.2557	780.45	4401014	786.08	794.4800	785.0200	794.02	762295
2017-01-06 00:00:00-05:00	782.28	799.4400	778.4800	795.99	4559445	795.26	807.9000	792.2041	806.12	967970
2017-01-09 00:00:00-05:00	798.00	801.7742	791.7700	796.92	2551340	806.40	809.9664	802.8300	806.58	777816

[6]:

```
df_stock_data_1
```

:[6]:

	AMZN					GOOG				
	open	high	low	close	volume	open	high	low	close	volume
time										
2019-01-02 00:00:00-05:00	1465.20	1553.36	1460.9300	1536.730	7132821	1016.57	1052.3200	1015.7100	1044.61	1184257
2019-01-03 00:00:00-05:00	1520.01	1538.00	1498.1062	1502.070	6340704	1041.00	1056.9800	1014.0800	1017.70	1381117
2019-01-04 00:00:00-05:00	1530.00	1594.00	1518.3100	1574.540	8285596	1033.00	1070.3000	1027.4178	1068.36	1629932
2019-01-07 00:00:00-05:00	1602.31	1634.56	1589.1850	1631.120	7252880	1071.50	1073.9999	1054.7600	1068.00	1599905
2019-01-08 00:00:00-05:00	1664.69	1676.61	1616.6100	1655.835	8184304	1076.11	1084.5600	1060.5300	1076.12	1301107
...	...	...	...	...	...	...	...	...	...	...

MC Simulation Code Slide 4

In [7]:

```
# Configuring a Monte Carlo simulation to forecast 3 years cumulative returns
MC_stocks_dist = MCSimulation(
    portfolio_data = df_stock_data,
    weights = [0.4, 0.6],
    num_simulation = 504,
    num_trading_days = 252*3
)
```

In [8]:

```
# Plot simulation outcomes
MC_stocks_dist.plot_simulation()
```

MC Simulation Summary Stats & Code

In [10]:
```
# Fetch summary statistics from the Monte Carlo simulation results
summary = MC_stocks_dist.summarize_cumulative_return()

# Print summary statistics
summary
```

Out[10]:
```
count           504.000000
mean              3.178783
std               1.155397
min               1.054909
25%               2.331218
50%               3.005301
75%               3.779923
max               8.076236
95% CI Lower      1.567885
95% CI Upper      5.993991
Name: 756, dtype: float64
```

In [11]:
```
# Set initial investment
initial_investment = 500000

# Use the lower and upper `95%` confidence intervals to calculate the range of the possible outcomes of our $500,0
ci_lower = round(summary[8]*initial_investment, 2)
ci_upper = round(summary[9]*initial_investment, 2)
# Print results
print(f"There is a 95% chance that an initial investment of ${initial_investment} in the portfoli
      f" over the next 3 years will end within in the range of"
      f" ${ci_lower} and ${ci_upper}")
```

```
There is a 95% chance that an initial investment of $500000 in the portfolio over the next 3 years wi
the range of $783942.27 and $2996995.27
```

MC (2nd) Simulation

In [12]:

```python
# Configuring a Monte Carlo simulation to forecast 3 years cumulative returns
MC_stocks_3 = MCSimulation(
    portfolio_data = df_stock_data,
    weights = [0.2, 0.8],
    num_simulation = 504,
    num_trading_days = 252*3
)
```

In [13]:

```python
# Running a Monte Carlo simulation to forecast 3 years cumulative returns
MC_stocks_3.calc_cumulative_return()
```

MC 2nd Simulation Code & Summary (con't)

In [16]:

```
# Fetch summary statistics from the Monte Carlo simulation results
summary_3 = MC_stocks_3.summarize_cumulative_return()

# Print summary statistics
summary_3
```

Out[16]:

```
count            504.000000
mean               3.034797
std                1.247481
min                0.825814
25%                2.187174
50%                2.836664
75%                3.653575
max                9.183340
95% CI Lower       1.207008
95% CI Upper       5.636321
Name: 756, dtype: float64
```

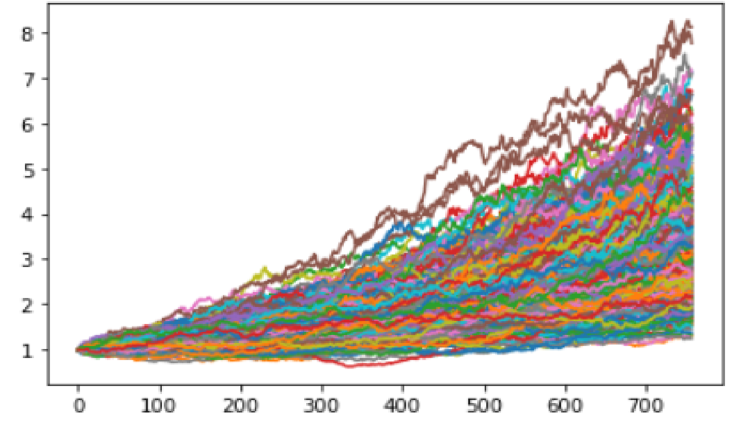
In [17]:

```
# Set initial investment
initial_investment = 500000

# Use the lower and upper `95%` confidence intervals to calculate the range of the possible outcomes of our $500,00
ci_lower_five = round(summary_3[8]*initial_investment, 2)
ci_upper_five = round(summary_3[9]*initial_investment, 2)

# Print results
print(f"There is a 95% chance that an initial investment of ${initial_investment} in the portfolio"
      f" over the next 3 years will end within in the range of"
      f" ${ci_lower_five} and ${ci_upper_five}")
```

```
There is a 95% chance that an initial investment of $500000 in the portfolio over the next 3 years wi
the range of $603504.19 and $2818160.71
```

MC Simulation 3 Year Growth

504 Simulations of Cumulative Portfolio Return Trajectories Over the Next 756 Trading Days.



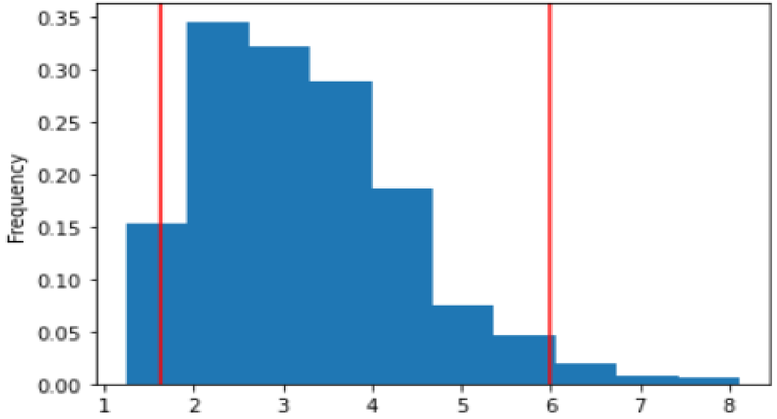
In [8]:

```
# Plot probability distribution and confidence intervals
MC_stocks_dist.plot_distribution()
```

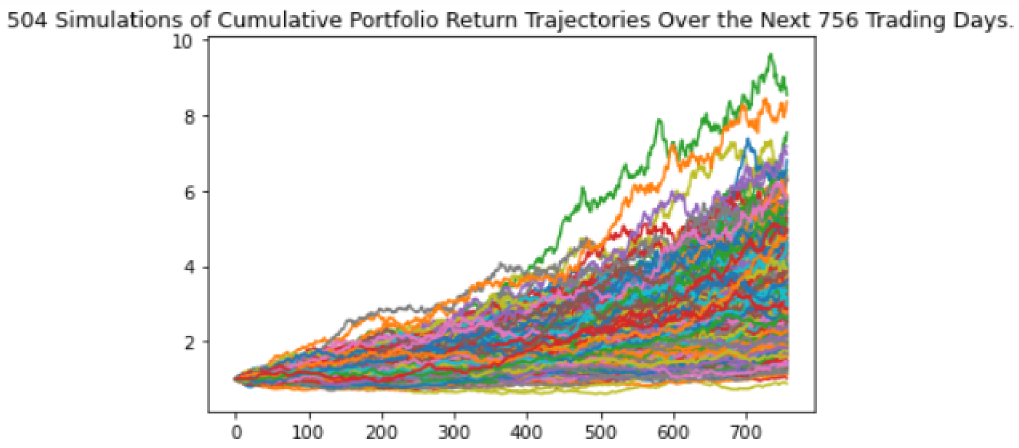
Out[8]:

```
<AxesSubplot:title={'center':'Distribution of Final Cumulative Returns Across All 504 Simulations'}, ylabel='Frequenc
y'>
```

Distribution of Final Cumulative Returns Across All 504 Simulations



MC Simulation Visualization 2

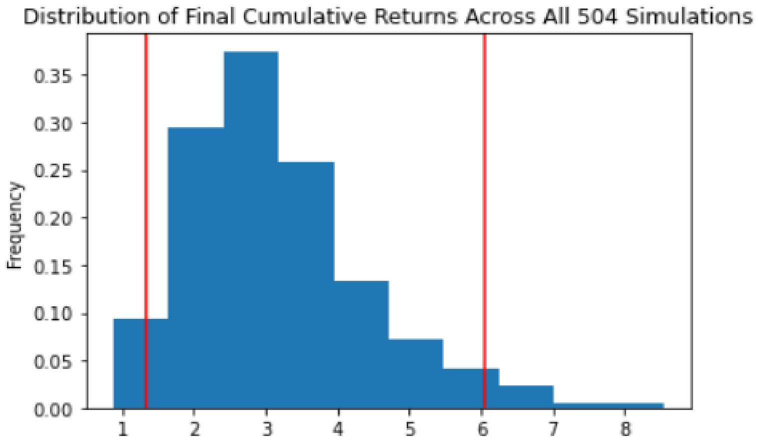


In [14]:

```python
# Plot probability distribution and confidence intervals
MC_stocks_3.plot_distribution()
```

Out[14]:

```
<AxesSubplot:title={'center':'Distribution of Final Cumulative Returns Across All 504 Simulations'}, ylabel='Frequenc
y'>
```



Machine Learning Random Forest Reg sklearn Code 1

In [1]:

```
import numpy as np
import pandas as pd
from pathlib import Path
from datetime import date
import matplotlib.pyplot as plt
import talib
```

In [2]:

```
from sklearn import metrics
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import ParameterGrid
```

In [3]:

```
import yfinance as yf
START = "2019-01-01"
TODAY = date.today().strftime("%Y-%m-%d")
ticker= "AMZN"
stock_data = yf.download(ticker, start=START, er
## preview data
stock_data
```

Machine Learning Random Forest Reg sklearn Code 2

In [6]:

```
feature_names = []
#for n in [14, 30, 50, 100, 200, 250]:
for n in [14,30,50]:
    stock_data['ma' + str(n)] = talib.SMA(stock_data['Close'].values, timeperiod=n)
    stock_data['rsi' + str(n)] = talib.RSI(stock_data['Close'].values, timeperiod=n)

    feature_names = feature_names + ['ma' + str(n), 'rsi' + str(n)]
```

In [7]:

```
stock_data['Volume_1d_change'] = stock_data['Volume'].pct_change()

volume_features = ['Volume_1d_change']
feature_names.extend(volume_features)
```

In [8]:

```
stock_data['5d_future_close'] = stock_data['Close'].shift(-5)
```

In [9]:

```
stock_data.dropna(inplace=True)
stock_data
```

Machine Learning Random Forest Reg sklearn Code 3

[75]:
```
#stock_data.dropna(inplace=True)

X = stock_data[feature_names]
y = stock_data['5d_future_close']

train_size = int(0.85 * y.shape[0])
X_train = X[:train_size]
y_train = y[:train_size]
X_test = X[train_size:]
y_test = y[train_size:]
```

[76]:
```
#grid = {'n_estimators': [200], 'max_depth': [3], 'max_features': [4, 8], 'random_state': [4
grid = {'n_estimators': [200], 'max_depth': [3], 'max_features': [4, 13], 'random_state': [4
test_scores = []

rf_model = RandomForestRegressor()

for g in ParameterGrid(grid):
    rf_model.set_params(**g)
    rf_model.fit(X_train, y_train)
    test_scores.append(rf_model.score(X_test, y_test))

best_index = np.argmax(test_scores)
print(test_scores[best_index], ParameterGrid(grid)[best_index])
```

```
0.2800579360364345 {'random_state': 42, 'n_estimators': 200, 'max_features': 13, 'max_d
3}
```

Machine Learning Random Forest Reg sklearn Code 4

In [77]:

```
print('Predicting base on: ')
print(feature_names)
```

```
Predicting base on:
['ma14', 'rsi14', 'ma30', 'rsi30', 'ma50', 'rsi50', 'ma100', 'rsi100', 'ma200', 'rsi200', 'ma
250', 'rsi250', 'Volume_1d_change']
```

In [78]:

```
print('MA: Moving Average', '\nRSI: Relative Strength Index')
```

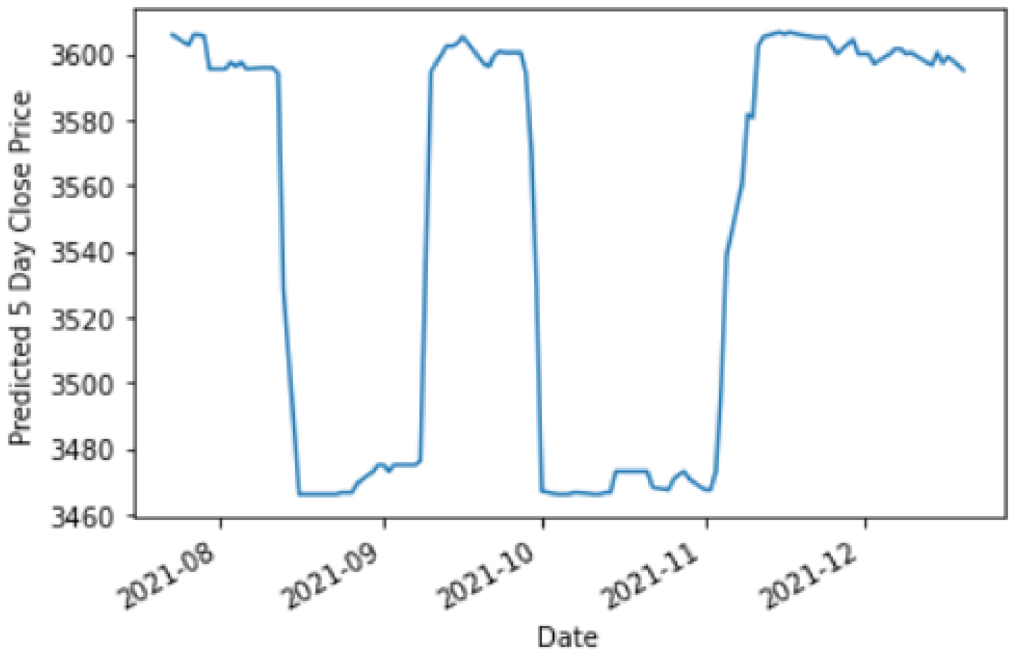
```
MA: Moving Average
RSI: Relative Strength Index
```

In [79]:

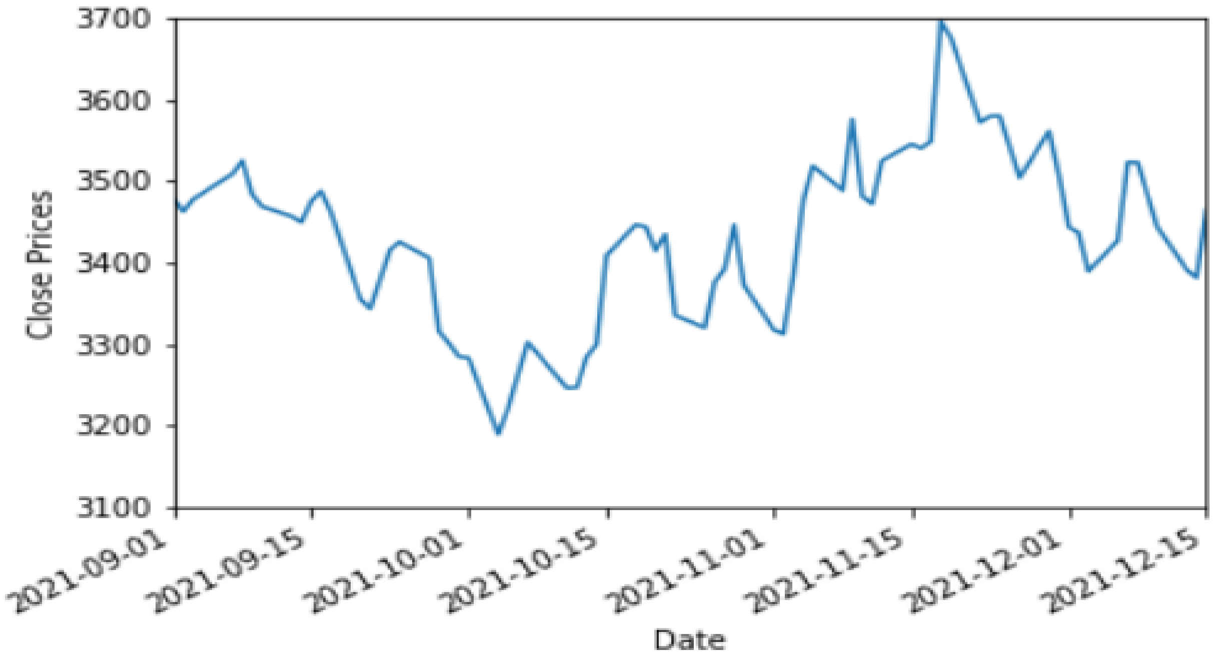
```
rf_model = RandomForestRegressor(n_estimators=200, max_depth=3, max_features=4, random_state
rf_model.fit(X_train, y_train)

y_pred = rf_model.predict(X_test)

y_pred_series = pd.Series(y_pred, index=y_test.index)
y_pred_series.plot()
plt.ylabel("Predicted 5 Day Close Price")
plt.show()
```

Machine Learning Random Forest Reg sklearn (round 1)

Predicted 5-Day Closing Price



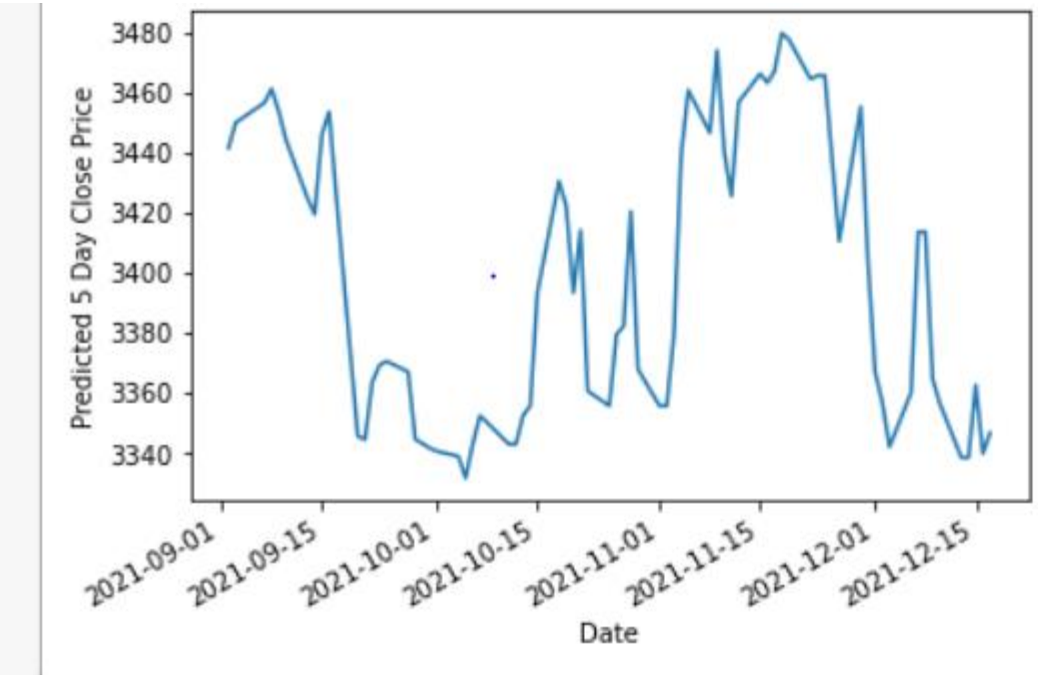
Actual Closing Prices



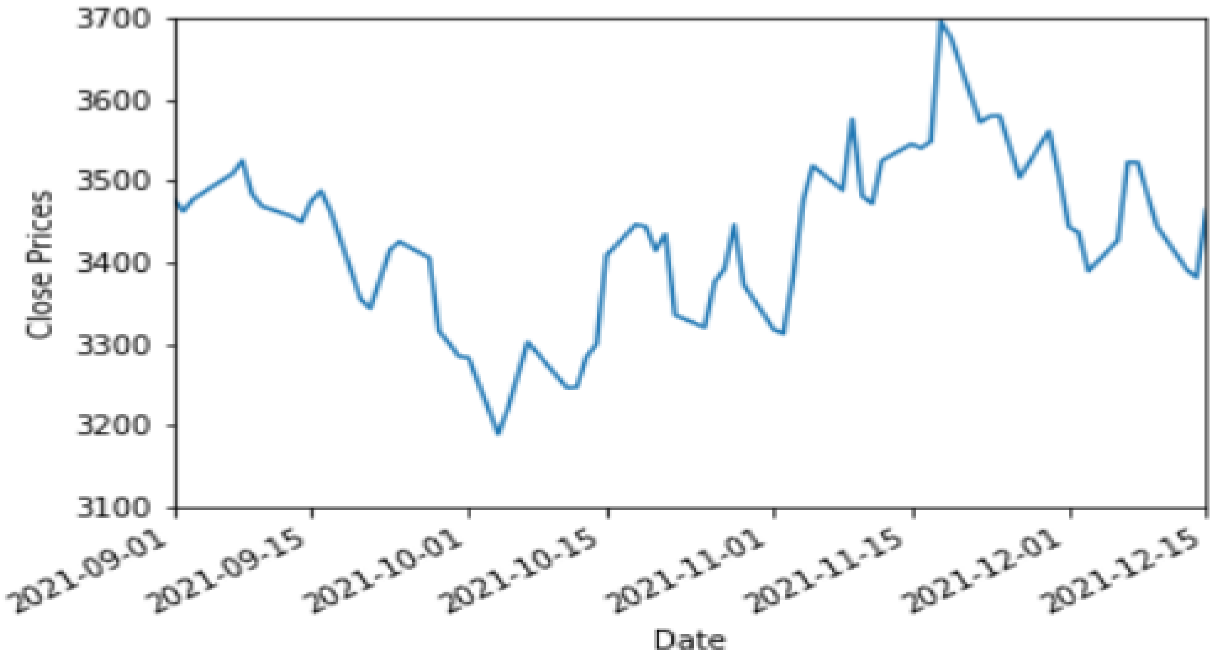
Model Mean Squared Error: 35,170

Machine Learning Random Forest Reg sklearn (final round)

Predicted 5-Day Closing Price

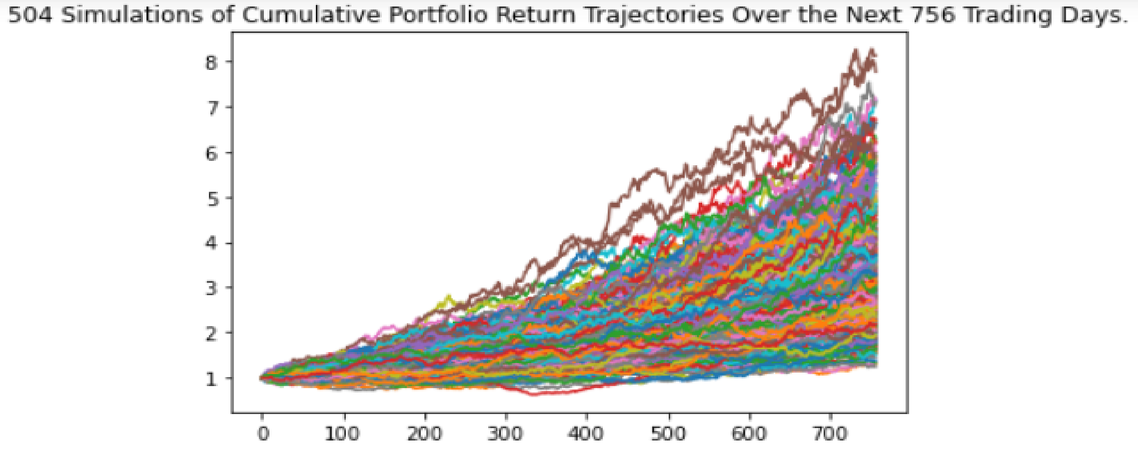


Actual Closing Prices



Model Mean Squared Error: 7,656

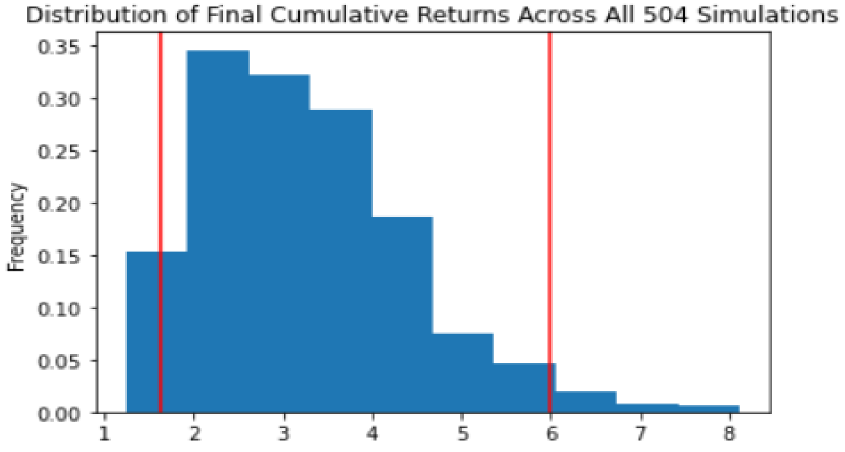
MC Simulation 3 Year Growth



In [8]:

```
# Plot probability distribution and confidence intervals
MC_stocks_dist.plot_distribution()
```

Out[8]: <AxesSubplot:title={'center':'Distribution of Final Cumulative Returns Across All 504 Simulations'}, ylabel='Frequency'>



Portfolio Python (P3)– Client's Portrait – Interview for Portfolio analysis - 2

Current Portfolio Composition	Proposed Portfolio Composition – Birthday Cake	Crypto Thoughts for future – Wedding Cake	Benchmarks
ADP : 1000 shares (from former employer from 24 yrs.)	AMZN, GOOG	BTC or ETH or SOL or DODG	SPY, NASD, DJIA
BR: 500 Shares			
CDK: 950 shares			
IRA value: 500K			

Conclusion & Recommendations - TDB

1. The current portfolio has performed well. ADP is a sound and growing Corporation. Its spin-offs helped diversify Anthon’s holdings.

2. Anthon’s IRA, managed by Morgan Stanley has seen healthy returns. Cupcake analyzed this portfolio.

3. The Anthon Portfolio can be replaced by a higher yielding Proposed Portfolio AMZN, GOOG. This provides strong results within the acceptable risk level of the investor.

4. Wedding Cake may include Cryptocurrencies.

Portfolio Python (P3)– Client’s Portrait – Interview for Portfolio analysis – Background Data

Demographics	Finance : current	Comments : Future
64 Years old	Annual Income: 100 K	Projected Social Security Monthly 2.8K
Married, 3 dependents	Home Value: 400K	Homes appreciating 2% annually
BS in Finance (has not yet studied Python nor pandas)	Mortgage owed: 200K Monthly payment $1,900	4-5 years into a 15 year mortgage @ 2.75%
Excellent Health	Annual Spending: 100K	Plays in league bowling and softball
Morgan Stanley Advisor(MS)	Savings: Stock, IRA, Pension	Largest asset IRA now managed by MS
Proposed retirement date : 11/1/2024	Monthly Mortgage: 1.9K	Retirement Net Worth Required : 1, 500K