CREDIT SUISSE GROUP AG (CS) - SNP500 - FTSE100

INSIGHTS INTO THE UK AND US ECONOMIES USING MACHINE LEARNING TECHNIQUES SUCH AS LINEAR REGRESSION AND NEURAL NETWORKS, BASED ON STOCK MARKET DATA.

Hemant Thapa

1. CREDIT SUISSE BACKGROUND

Credit Suisse is a Swiss multinational investment bank and financial services company that was founded in 1856. The company has a long history of providing banking and financial services to individuals, corporations, and governments around the world.

In recent years, Credit Suisse has been facing significant financial problems, which have put pressure on the global banking system. One of the key issues has been the recent takeover of Credit Suisse by its larger rival UBS, which resulted in the investment of bondholders being wiped out. The bondholders who owned about $17 billion of risky Credit Suisse debt were left with nothing after the takeover.

The problem at Credit Suisse centers around the hierarchy of who should lose out first when a bank struggles. On one hand, investors who bought risky AT1 bonds (Additional Tier 1 bank debt) should expect to lose money when a bank runs into trouble. However, there are questions because Credit Suisse's shareholders will not be wiped out completely but are set to be compensated in the emergency takeover with UBS shares worth the equivalent of Sfr 0.76 (£0.67) a share.

AT1 bonds, also known as Additional Tier 1 bonds, are a type of debt instrument issued by banks to raise capital. They are hybrid securities that combine characteristics of both bonds and equity, and are designed to increase the safety buffers of banks in times of financial stress. If a bank's capital levels fall below a certain level, the AT1 bonds can be converted into equity, which helps to reduce the bank's debts and increase its capitalization. These bonds were introduced after the 2008 financial crisis as a way to "bail in" failing banks, by imposing losses on investors rather than relying on taxpayer-funded bailouts. As a result, AT1 bonds are considered riskier than other types of debt, and investors are typically offered a higher return to compensate for this risk.

In [ ]:
```
import yfinance as yf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import requests
import lxml
import appdirs
import pytz
import frozendict
from bs4 import BeautifulSoup
import html5lib
import pandas_datareader as pdr
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
import tensorflow as tf
import keras
from keras.models import Sequential
from keras.layers import Dense
import torch
import xgboost as xgb
import ta
import warnings
warnings.filterwarnings("ignore")
```

To assess the financial stability of companies like Credit Suisse, we can perform various analyses using tools like Python. For example, we can use Python to perform a financial ratio analysis, which would involve calculating various ratios such as the debt-to-equity ratio, the current ratio, and the return on equity. This analysis would give us a good understanding of the company's financial health and help us determine whether it is likely to face any financial problems in the future.

Another useful tool for predicting the financial stability of companies is machine learning algorithms. For example, we could use algorithms like decision trees, random forests, and neural networks to analyze the financial data of companies and make predictions about their future performance. These algorithms could be trained on historical financial data and used to identify patterns and trends in the data. This would allow us to make more informed decisions about investing in companies like Credit Suisse.

2. U.S MARKET

The S&P 500 can have a significant impact on the global economy due to the sheer size and influence of the US economy, which is the largest in the world. The performance of the S&P 500 can affect not only the US economy but also other countries and regions that are closely tied to the US economy through trade, investment, and other forms of economic activity.

When the S&P 500 performs well, it can boost investor confidence and lead to increased investment in the US economy, which can, in turn, stimulate economic growth and job creation. A strong US economy can also increase demand for goods and services from other countries, thereby supporting economic growth around the world.

Conversely, when the S&P 500 performs poorly, it can have negative ripple effects on the global economy. A decline in the US stock market can reduce investor confidence and lead to a reduction in investment and economic activity, both in the US and in other countries that are closely tied to the US economy.

Moreover, the S&P 500 can serve as a bellwether for broader trends in the global economy. Changes in the index can reflect shifts in consumer behavior, changes in the regulatory environment, and other factors that can have far-reaching impacts on the global economy.

In [2]:
```
df = pd.read_csv('https://pkgstore.datahub.io/core/s-and-p-500-companies/
df.tail()
```

Out[2]:

	Symbol	Name	Sector
500	YUM	Yum! Brands	Consumer Discretionary
501	ZBRA	Zebra Technologies	Information Technology
502	ZBH	Zimmer Biomet	Health Care
503	ZION	Zions Bancorp	Financials
504	ZTS	Zoetis	Health Care

In [3]:
```
df.to_csv('SNP500.csv')
```

In [4]:
```
df.columns = df.columns.str.lower()
companies = df.name.unique()
companies
```

Out[4]:
```
array(['3M', 'A. O. Smith', 'Abbott Laboratories', 'AbbVie', 'Abiomed',
       'Accenture', 'Activision Blizzard', 'ADM', 'Adobe',
       'Advance Auto Parts', 'Advanced Micro Devices', 'AES Corp',
       'Aflac', 'Agilent Technologies', 'Air Products & Chemicals',
       'Akamai Technologies', 'Alaska Air Group', 'Albemarle Corporation
',
       'Alexandria Real Estate Equities', 'Align Technology', 'Allegion'
,
       'Alliant Energy', 'Allstate Corp', 'Alphabet (Class A)',
       'Alphabet (Class C)', 'Altria Group', 'Amazon', 'Amcor',
       'Ameren Corp', 'American Airlines Group',
       'American Electric Power', 'American Express',
       'American International Group', 'American Tower',
       'American Water Works', 'Ameriprise Financial',
       'AmerisourceBergen', 'Ametek', 'Amgen', 'Amphenol',
       'Analog Devices', 'Ansys', 'Anthem', 'Aon', 'APA Corporation',
       'Apple', 'Applied Materials', 'Aptiv', 'Arista Networks',
       'Arthur J. Gallagher & Co.', 'Assurant', 'AT&T', 'Atmos Energy',
       'Autodesk', 'Automatic Data Processing', 'AutoZone',
       'AvalonBay Communities', 'Avery Dennison', 'Baker Hughes',
       'Ball Corp', 'Bank of America', 'Bath & Body Works Inc.',
       'Baxter International', 'Becton Dickinson', 'Berkshire Hathaway',
       'Best Buy', 'Bio-Rad Laboratories', 'Bio-Techne', 'Biogen',
       'BlackRock', 'BNY Mellon', 'Boeing', 'Booking Holdings',
       'BorgWarner', 'Boston Properties', 'Boston Scientific',
       'Bristol Myers Squibb', 'Broadcom',
       'Broadridge Financial Solutions', 'Brown & Brown', 'Brown-Forman'
,
       'C. H. Robinson', 'Cadence Design Systems',
       'Caesars Entertainment', 'Campbell Soup', 'Capital One Financial'
,
       'Cardinal Health', 'CarMax', 'Carnival Corporation',
       'Carrier Global', 'Catalent', 'Caterpillar', 'Cboe Global Markets
',
       'CBRE', 'CDW', 'Celanese', 'Centene Corporation',
       'CenterPoint Energy', 'Ceridian', 'Cerner', 'CF Industries',
       'Charles River Laboratories', 'Charles Schwab Corporation',
       'Charter Communications', 'Chevron Corporation',
       'Chipotle Mexican Grill', 'Chubb', 'Church & Dwight', 'Cigna',
       'Cincinnati Financial', 'Cintas Corporation', 'Cisco Systems',
       'Citigroup', 'Citizens Financial Group', 'Citrix Systems',
       'Clorox', 'CME Group', 'CMS Energy', 'Coca-Cola Company',
       'Cognizant Technology Solutions', 'Colgate-Palmolive', 'Comcast',
       'Comerica', 'Conagra Brands', 'ConocoPhillips',
       'Consolidated Edison', 'Constellation Brands', 'Copart', 'Corning
',
       'Corteva', 'Costco', 'Coterra', 'Crown Castle', 'CSX', 'Cummins',
       'CVS Health', 'D. R. Horton', 'Danaher Corporation',
       'Darden Restaurants', 'DaVita', 'Deere & Co.', 'Delta Air Lines',
       'Dentsply Sirona', 'Devon Energy', 'DexCom', 'Diamondback Energy'
,
       'Digital Realty Trust', 'Discover Financial Services',
```

```
        'Discovery (Series A)', 'Discovery (Series C)', 'Dish Network',
        'Dollar General', 'Dollar Tree', 'Dominion Energy',
        "Domino's Pizza", 'Dover Corporation', 'Dow', 'DTE Energy',
        'Duke Energy', 'Duke Realty Corp', 'DuPont', 'DXC Technology',
        'Eastman Chemical', 'Eaton Corporation', 'eBay', 'Ecolab',
        'Edison International', 'Edwards Lifesciences', 'Electronic Arts'
    ,
        'Eli Lilly & Co', 'Emerson Electric Company', 'Enphase Energy',
        'Entergy', 'EOG Resources', 'Equifax', 'Equinix',
        'Equity Residential', 'Essex Property Trust',
        'Estée Lauder Companies', 'Etsy', 'Everest Re', 'Evergy',
        'Eversource Energy', 'Exelon', 'Expedia Group', 'Expeditors',
        'Extra Space Storage', 'ExxonMobil', 'F5 Networks', 'Facebook',
        'Fastenal', 'Federal Realty Investment Trust', 'FedEx',
        'Fidelity National Information Services', 'Fifth Third Bancorp',
        'First Republic Bank', 'FirstEnergy', 'Fiserv', 'Fleetcor',
        'FMC Corporation', 'Ford', 'Fortinet', 'Fortive',
        'Fortune Brands Home & Security', 'Fox Corporation (Class A)',
        'Fox Corporation (Class B)', 'Franklin Resources',
        'Freeport-McMoRan', 'Gap', 'Garmin', 'Gartner', 'Generac Holdings
    ',
        'General Dynamics', 'General Electric', 'General Mills',
        'General Motors', 'Genuine Parts', 'Gilead Sciences',
        'Global Payments', 'Globe Life', 'Goldman Sachs', 'Halliburton',
        'Hanesbrands', 'Hasbro', 'HCA Healthcare', 'Healthpeak Properties
    ',
        'Henry Schein', 'Hess Corporation', 'Hewlett Packard Enterprise',
        'Hilton Worldwide', 'Hologic', 'Home Depot', 'Honeywell', 'Hormel
    ',
        'Host Hotels & Resorts', 'Howmet Aerospace', 'HP', 'Humana',
        'Huntington Bancshares', 'Huntington Ingalls Industries', 'IBM',
        'IDEX Corporation', 'Idexx Laboratories', 'IHS Markit',
        'Illinois Tool Works', 'Illumina', 'Incyte', 'Ingersoll Rand',
        'Intel', 'Intercontinental Exchange',
        'International Flavors & Fragrances', 'International Paper',
        'Interpublic Group', 'Intuit', 'Intuitive Surgical', 'Invesco',
        'IPG Photonics', 'IQVIA', 'Iron Mountain', 'J. B. Hunt',
        'Jack Henry & Associates', 'Jacobs Engineering Group',
        'JM Smucker', 'Johnson & Johnson', 'Johnson Controls',
        'JPMorgan Chase', 'Juniper Networks', 'Kansas City Southern',
        "Kellogg's", 'KeyCorp', 'Keysight Technologies', 'Kimberly-Clark'
    ,
        'Kimco Realty', 'Kinder Morgan', 'KLA Corporation', 'Kraft Heinz'
    ,
        'Kroger', 'L3Harris Technologies', 'LabCorp', 'Lam Research',
        'Lamb Weston', 'Las Vegas Sands', 'Leggett & Platt', 'Leidos',
        'Lennar', 'Lincoln National', 'Linde', 'Live Nation Entertainment
    ',
        'LKQ Corporation', 'Lockheed Martin', 'Loews Corporation',
        "Lowe's", 'Lumen Technologies', 'LyondellBasell', 'M&T Bank',
        'Marathon Oil', 'Marathon Petroleum', 'MarketAxess',
        'Marriott International', 'Marsh & McLennan',
        'Martin Marietta Materials', 'Masco', 'Mastercard', 'Match Group'
    ,
        'McCormick & Company', "McDonald's", 'McKesson Corporation',
        'Medtronic', 'Merck & Co.', 'MetLife', 'Mettler Toledo',
        'MGM Resorts International', 'Microchip Technology',
```

```
        'Micron Technology', 'Microsoft', 'Mid-America Apartments',
        'Moderna', 'Mohawk Industries', 'Molson Coors Beverage Company',
        'Mondelez International', 'Monolithic Power Systems',
        'Monster Beverage', "Moody's Corporation", 'Morgan Stanley',
        'Motorola Solutions', 'MSCI', 'Nasdaq', 'NetApp', 'Netflix',
        'Newell Brands', 'Newmont', 'News Corp (Class A)',
        'News Corp (Class B)', 'NextEra Energy', 'Nielsen Holdings',
        'Nike', 'NiSource', 'Norfolk Southern', 'Northern Trust',
        'Northrop Grumman', 'NortonLifeLock',
        'Norwegian Cruise Line Holdings', 'NRG Energy', 'Nucor', 'Nvidia'
    ,
        'NVR', 'NXP', "O'Reilly Automotive", 'Occidental Petroleum',
        'Old Dominion Freight Line', 'Omnicom Group', 'Oneok', 'Oracle',
        'Organon & Co.', 'Otis Worldwide', 'Paccar',
        'Packaging Corporation of America', 'Parker-Hannifin', 'Paychex',
        'Paycom', 'PayPal', 'Penn National Gaming', 'Pentair',
       "People's United Financial", 'PepsiCo', 'PerkinElmer', 'Pfizer',
        'Philip Morris International', 'Phillips 66',
        'Pinnacle West Capital', 'Pioneer Natural Resources',
        'PNC Financial Services', 'Pool Corporation', 'PPG Industries',
        'PPL', 'Principal Financial Group', 'Procter & Gamble',
        'Progressive Corporation', 'Prologis', 'Prudential Financial',
        'PTC', 'Public Service Enterprise Group', 'Public Storage',
        'PulteGroup', 'PVH', 'Qorvo', 'Qualcomm', 'Quanta Services',
        'Quest Diagnostics', 'Ralph Lauren Corporation',
        'Raymond James Financial', 'Raytheon Technologies',
        'Realty Income Corporation', 'Regency Centers',
        'Regeneron Pharmaceuticals', 'Regions Financial Corporation',
        'Republic Services', 'ResMed', 'Robert Half International',
        'Rockwell Automation', 'Rollins', 'Roper Technologies',
        'Ross Stores', 'Royal Caribbean Group', 'S&P Global', 'Salesforce
    ',
        'SBA Communications', 'Schlumberger', 'Seagate Technology',
        'Sealed Air', 'Sempra Energy', 'ServiceNow', 'Sherwin-Williams',
        'Simon Property Group', 'Skyworks Solutions', 'Snap-on',
        'Southern Company', 'Southwest Airlines', 'Stanley Black & Decker
    ',
        'Starbucks', 'State Street Corporation', 'Steris',
        'Stryker Corporation', 'SVB Financial', 'Synchrony Financial',
        'Synopsys', 'Sysco', 'T-Mobile US', 'T. Rowe Price',
        'Take-Two Interactive', 'Tapestry', 'Target Corporation',
        'TE Connectivity', 'Teledyne Technologies', 'Teleflex', 'Teradyne
    ',
        'Tesla', 'Texas Instruments', 'Textron', 'The Cooper Companies',
        'The Hartford', 'The Hershey Company', 'The Mosaic Company',
        'The Travelers Companies', 'The Walt Disney Company',
        'Thermo Fisher Scientific', 'TJX Companies',
        'Tractor Supply Company', 'Trane Technologies', 'TransDigm Group'
    ,
        'Trimble', 'Truist Financial', 'Twitter', 'Tyler Technologies',
        'Tyson Foods', 'U.S. Bancorp', 'UDR', 'Ulta Beauty',
        'Under Armour (Class A)', 'Under Armour (Class C)',
        'Union Pacific', 'United Airlines', 'United Parcel Service',
        'United Rentals', 'UnitedHealth Group',
        'Universal Health Services', 'Valero Energy', 'Ventas', 'Verisign
    ',
        'Verisk Analytics', 'Verizon Communications',
```

```
       'Vertex Pharmaceuticals', 'VF Corporation', 'ViacomCBS', 'Viatris
',
       'Visa', 'Vornado Realty Trust', 'Vulcan Materials',
       'W. R. Berkley Corporation', 'W. W. Grainger', 'Wabtec',
       'Walgreens Boots Alliance', 'Walmart', 'Waste Management',
       'Waters Corporation', 'WEC Energy Group', 'Wells Fargo',
       'Welltower', 'West Pharmaceutical Services', 'Western Digital',
       'Western Union', 'WestRock', 'Weyerhaeuser',
       'Whirlpool Corporation', 'Williams Companies',
       'Willis Towers Watson', 'Wynn Resorts', 'Xcel Energy', 'Xilinx',
       'Xylem', 'Yum! Brands', 'Zebra Technologies', 'Zimmer Biomet',
       'Zions Bancorp', 'Zoetis'], dtype=object)
```

In [5]:
```
us_500 = []
ticker = df.symbol.unique()
for i in ticker:
    us_500.append(i)
```

In [6]:
```
us_500[:10]
```

Out[6]:
```
['MMM', 'AOS', 'ABT', 'ABBV', 'ABMD', 'ACN', 'ATVI', 'ADM', 'ADBE', 'AAP
']
```

In [7]:
```
# Load the data into a pandas DataFrame
us_price_data = yf.download(us_500, start="2023-03-13", end="2023-03-20")
us_price_data.dropna(axis=1, inplace=True)
```

```
[**********100%**********]  505 of 505 completed

21 Failed downloads:
- NLOK: No timezone found, symbol may be delisted
- DISCK: No timezone found, symbol may be delisted
- VIAC: No timezone found, symbol may be delisted
- FBHS: No timezone found, symbol may be delisted
- KSU: No timezone found, symbol may be delisted
- BF.B: No data found for this date range, symbol may be delisted
- BRK.B: No timezone found, symbol may be delisted
- BLL: No timezone found, symbol may be delisted
- PBCT: No timezone found, symbol may be delisted
- WLTW: No timezone found, symbol may be delisted
- DRE: No timezone found, symbol may be delisted
- NLSN: No timezone found, symbol may be delisted
- ABMD: No data found for this date range, symbol may be delisted
- TWTR: No data found, symbol may be delisted
- FB: No timezone found, symbol may be delisted
- CERN: No timezone found, symbol may be delisted
- ANTM: No timezone found, symbol may be delisted
- INFO: No timezone found, symbol may be delisted
- DISCA: No timezone found, symbol may be delisted
- CTXS: No timezone found, symbol may be delisted
- XLNX: No timezone found, symbol may be delisted
```

In [8]:
```
us_price_data.shape
```

Out[8]:
```
(5, 484)
```

In [9]:
```
df_weekly = pd.DataFrame(data=us_price_data)
df_weekly.columns = df_weekly.columns.str.title()
df_weekly
```

Out[9]:

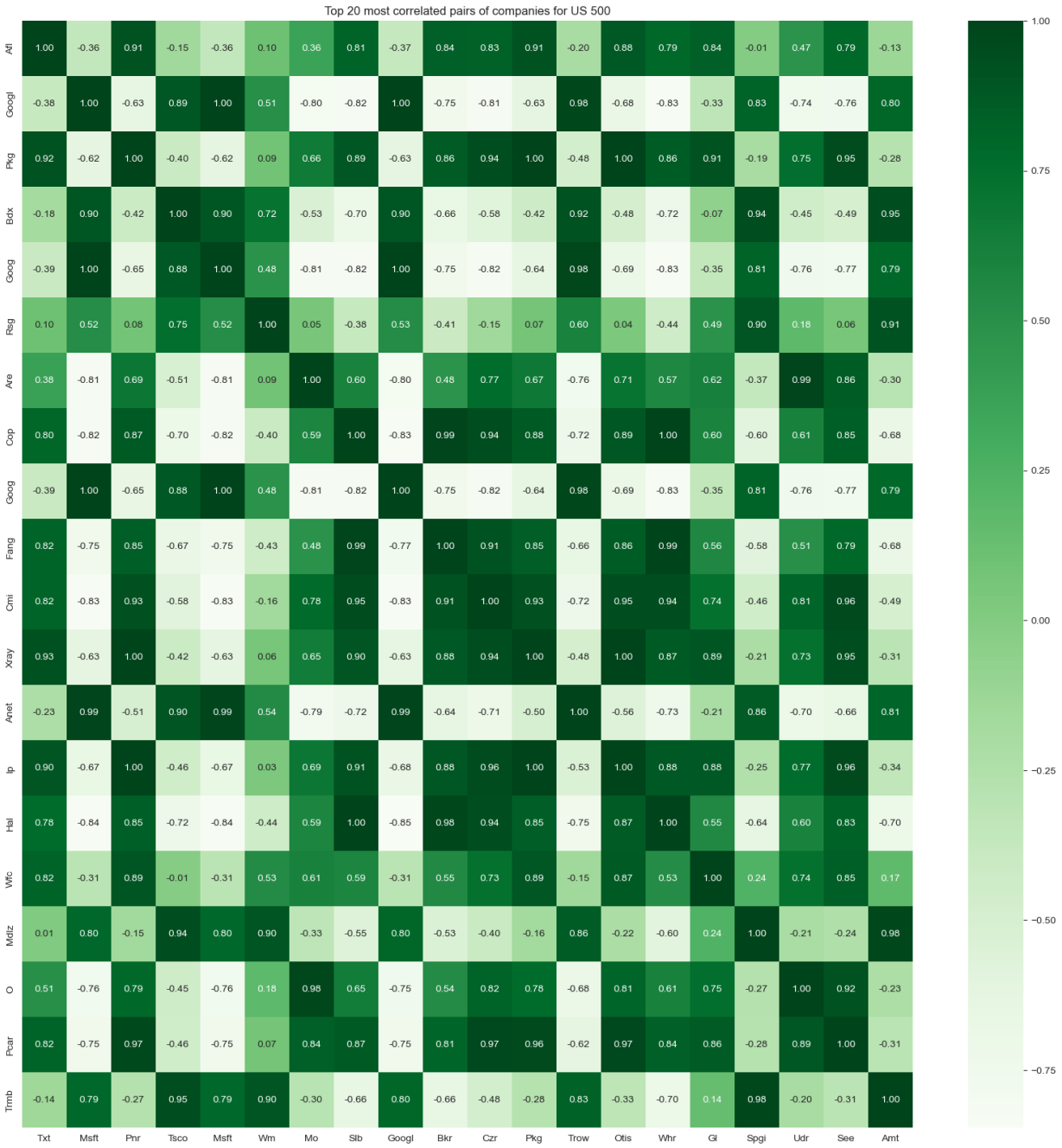
Date	A	Aal	Aap	Aapl	Abbv	Abc	Al
2023-03-13 00:00:00	136.690002	14.85	124.360001	150.470001	151.949997	148.369995	98.19000
2023-03-14 00:00:00	138.399994	14.66	121.699997	152.589996	153.850006	149.289993	98.55000
2023-03-15 00:00:00	134.029999	13.86	121.769997	152.990005	154.059998	149.610001	97.80000
2023-03-16 00:00:00	136.820007	14.12	122.029999	155.850006	155.300003	152.050003	99.02999
2023-03-17 00:00:00	133.179993	13.98	119.400002	155.000000	154.220001	152.020004	97.01000

5 rows × 484 columns

ONE WEEK STOCK PRICE CORRELATION ANALYSIS AMONG TOP 20 US COMPANIES

In [10]:
```
# Compute the correlation matrix
corr_matrix = us_price_data.corr()

# Find the top 20 most correlated pairs of companies
corr_pairs = corr_matrix.unstack().sort_values(ascending=False)
corr_pairs = corr_pairs[corr_pairs != 1].drop_duplicates()[:20]
corr_pairs = pd.DataFrame(corr_pairs).reset_index()
```

In [11]:
```
cmap = 'Greens'
sns.set_style("white")
plt.figure(figsize=(20, 20))
sns.heatmap(corr_matrix.loc[corr_pairs.iloc[:,0], corr_pairs.iloc[:,1]],
plt.title("Top 20 most correlated pairs of companies for US 500")
plt.show()
```



ANALYSIS OF SIX MONTHS DATA SET OF TOP US 500 COMPANIES

In [12]:
```
us_price_data = yf.download(us_500, start="2022-09-21", end="2023-03-21")
us_price_data.dropna(axis=1, inplace=True)
```

```
[**********100%**********]  505 of 505 completed

20 Failed downloads:
- NLOK: No timezone found, symbol may be delisted
- DISCK: No timezone found, symbol may be delisted
- VIAC: No timezone found, symbol may be delisted
- FBHS: No timezone found, symbol may be delisted
- KSU: No timezone found, symbol may be delisted
- BF.B: No data found for this date range, symbol may be delisted
- BRK.B: No timezone found, symbol may be delisted
- BLL: No timezone found, symbol may be delisted
- PBCT: No timezone found, symbol may be delisted
- WLTW: No timezone found, symbol may be delisted
- DRE: No timezone found, symbol may be delisted
- NLSN: No timezone found, symbol may be delisted
- TWTR: No data found, symbol may be delisted
- FB: No timezone found, symbol may be delisted
- CERN: No timezone found, symbol may be delisted
- ANTM: No timezone found, symbol may be delisted
- INFO: No timezone found, symbol may be delisted
- DISCA: No timezone found, symbol may be delisted
- CTXS: No timezone found, symbol may be delisted
- XLNX: No timezone found, symbol may be delisted
```

In [13]:
```
us_price_data[:5]
```

Out[13]:

Date	A	AAL	AAP	AAPL	ABBV	ABC	A
2022-09-21 00:00:00	126.480003	13.23	165.500000	153.720001	137.588730	139.119995	100.6500
2022-09-22 00:00:00	124.389999	12.71	164.110001	152.740005	140.236374	140.009995	101.0700
2022-09-23 00:00:00	123.480003	12.21	160.500000	150.429993	140.285400	137.199997	100.6800
2022-09-26 00:00:00	122.309998	11.86	157.679993	150.770004	138.471298	136.350006	99.8399
2022-09-27 00:00:00	121.610001	12.27	159.309998	151.759995	138.971390	135.440002	98.3300

5 rows × 484 columns

In [14]:
```
us_price_data.shape
```

Out[14]:
```
(124, 484)
```

TOP 20 LESS VOLATILE COMPANIES IN SNP500 (LAST 6 MONTHS)

Now we are building an algortihms for seprating top less volatile companies for investment.

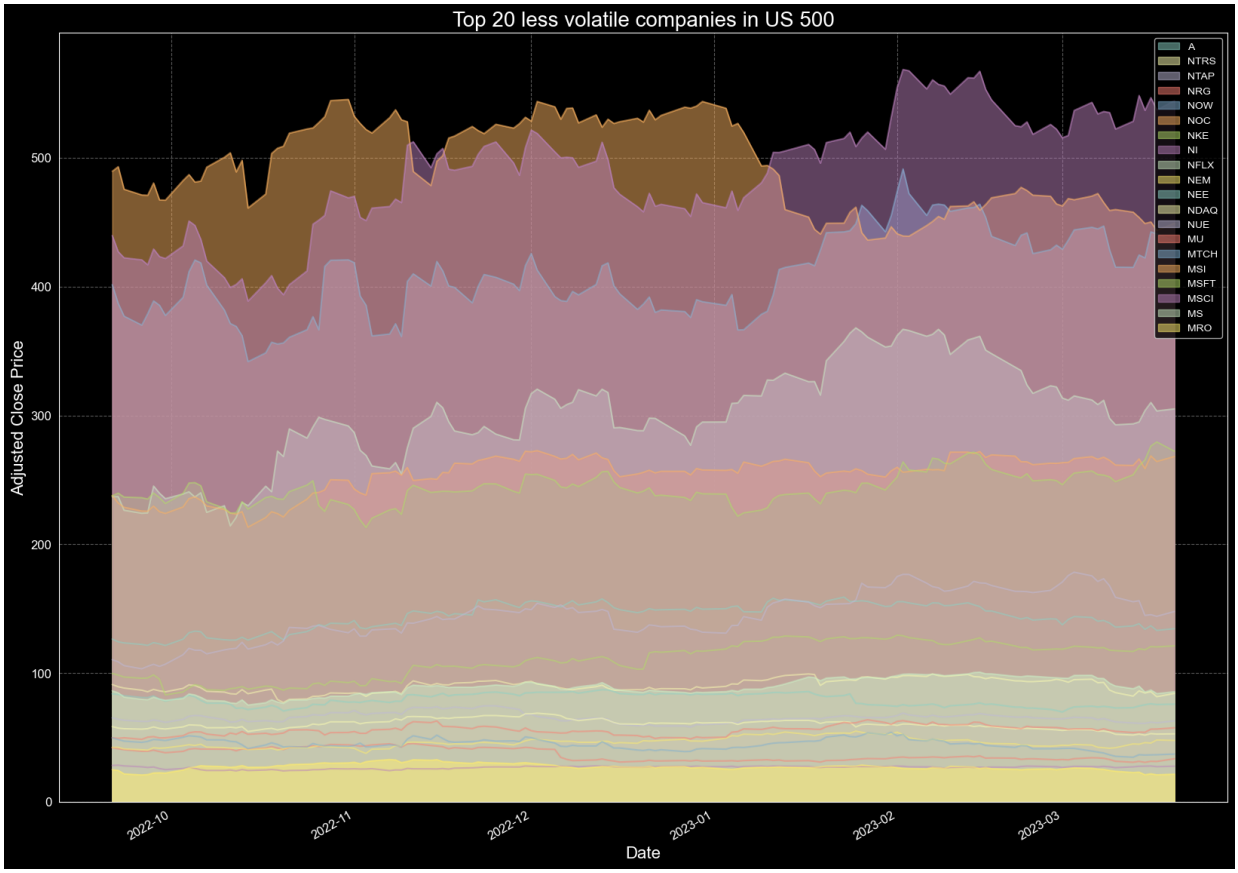
This code is using technical analysis to identify less volatile companies in the SNP500 index. It calculates the relative strength index (RSI) for each company's stock prices using the ta.momentum.RSIIndicator function from the ta library. It then filters the companies where the RSI is below 30, which is considered oversold and potentially undervalued. Finally, it selects the top 20 less volatile companies based on the RSI and sorts them in ascending order using the sort_values method. The resulting companies are stored in the safe_companies variable.

In [32]:
```
#less volatile companies based on technical analysis
rsi_data = pd.DataFrame()
for col in us_price_data.columns:
    rsi = ta.momentum.RSIIndicator(us_price_data[col]).rsi()
    rsi_data[col] = rsi

#companies with RSI < 30
low_rsi = rsi_data.iloc[-1] < 30
```

In [33]:
```
#top 20 less volatile companies
safe_companies = low_rsi.sort_values().index[:20]
```

In [34]:
```
# Plot the prices of the top 20 safe companies as an area graph
plt.style.use('dark_background')
plt.figure(figsize=(20,6))
us_price_data[safe_companies].plot(kind='area', alpha=0.5, stacked=False,
plt.title("Top 20 less volatile companies in US 500", fontsize=20)
plt.xlabel("Date", fontsize=16)
plt.ylabel("Adjusted Close Price", fontsize=16)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.grid(linestyle='--', color='gray', alpha=0.7)
plt.show()
plt.close()
```

```
<Figure size 2000x600 with 0 Axes>
```

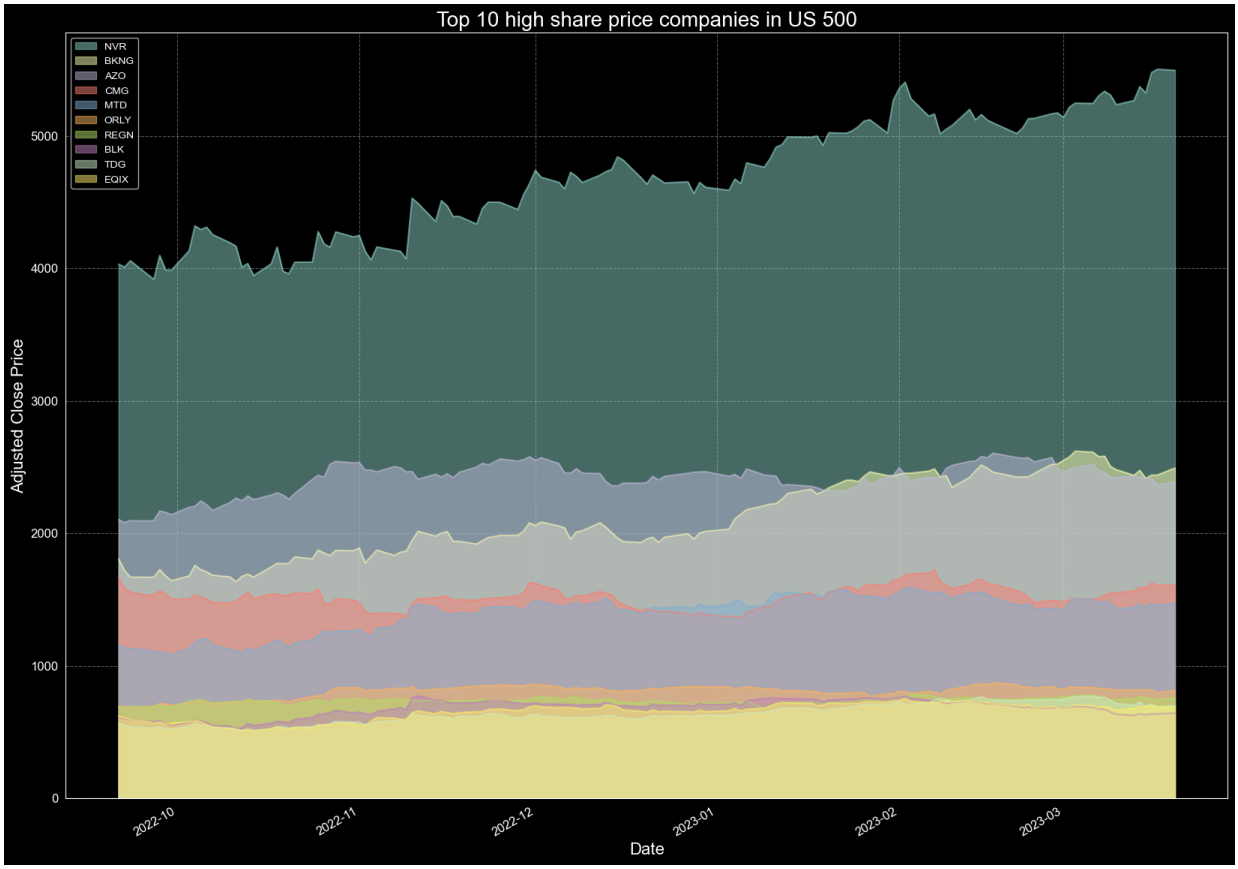


TOP 10 HIGH SHARE PRICE AND LOW SHARE PRICE IN SNP500 (IAST SIX MONTHS)

In [35]:
```
high_price_companies = us_price_data.max().sort_values(ascending=False).i
low_price_companies = us_price_data.max().sort_values().index[:10]
```

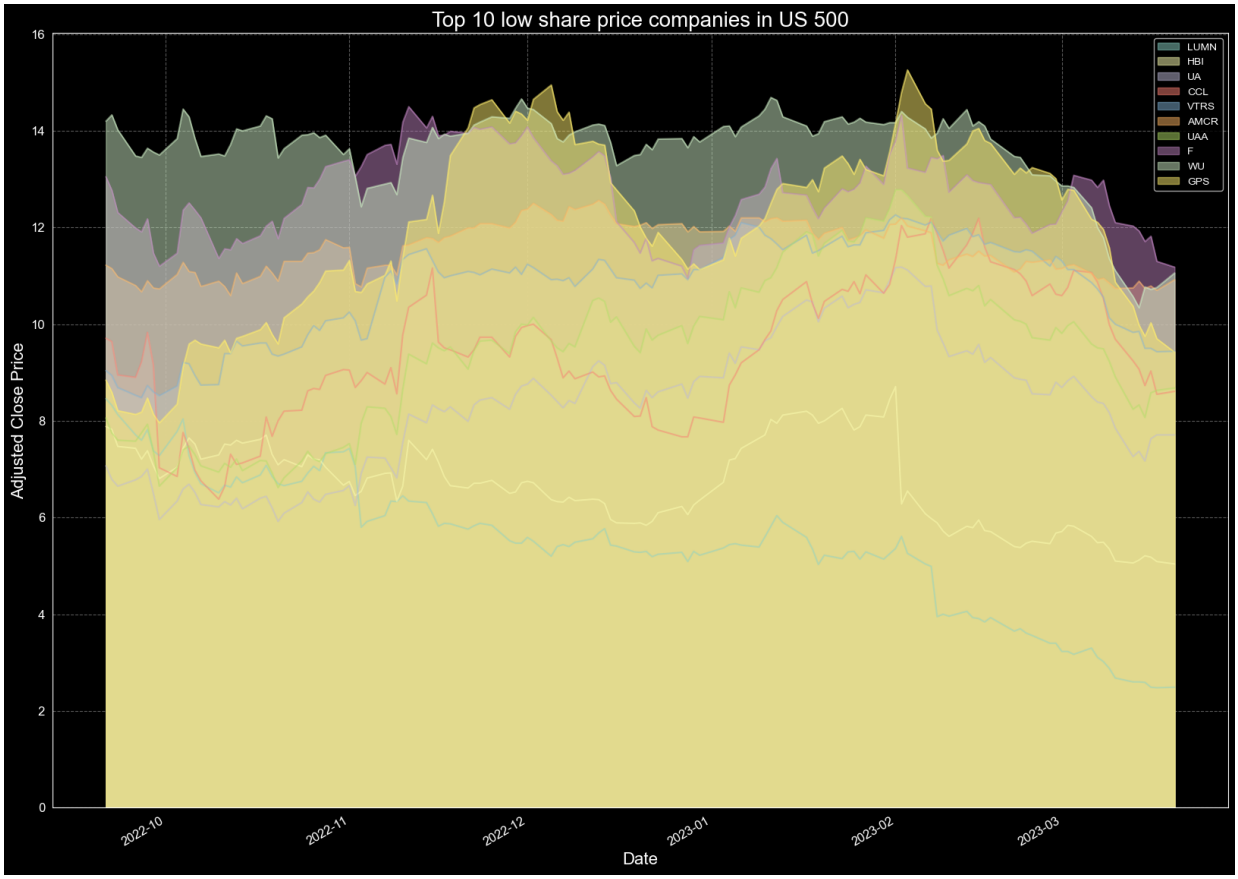
In [36]:
```
#top high share price companies as an area graph
plt.style.use('dark_background')
plt.figure(figsize=(20,6))
us_price_data[high_price_companies].plot(kind='area', alpha=0.5, stacked=
plt.title("Top 10 high share price companies in US 500", fontsize=20)
plt.xlabel("Date", fontsize=16)
plt.ylabel("Adjusted Close Price", fontsize=16)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.grid(linestyle='--', color='gray', alpha=0.7)
plt.show()
plt.close()
```

```
<Figure size 2000x600 with 0 Axes>
```



In [37]:
```
# Plot the prices of top low share price companies as an area graph
plt.style.use('dark_background')
plt.figure(figsize=(20,6))
us_price_data[low_price_companies].plot(kind='area', alpha=0.5, stacked=F
plt.title("Top 10 low share price companies in US 500", fontsize=20)
plt.xlabel("Date", fontsize=16)
plt.ylabel("Adjusted Close Price", fontsize=16)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.grid(linestyle='--', color='gray', alpha=0.7)
plt.show()
plt.close()
```

```
<Figure size 2000x600 with 0 Axes>
```



3. U.K MARKET

The FTSE 100 is a stock market index that gauges the performance of the 100 largest publicly-traded companies in the United Kingdom. It is widely considered as a top indicator of the UK stock market's health and the overall UK economy.

The FTSE 100 also has a significant influence on the global economy since the UK is among the largest economies in the world and a key player in global trade and investment. The index's performance can impact not only the UK economy but also other nations and regions closely associated with the UK economy through trade, investment, and other forms of economic activity.

When the FTSE 100 performs well, it can foster investor confidence and lead to increased investment in the UK economy. This can, in turn, fuel economic growth and job creation. A robust UK economy can also increase demand for goods and services from other countries, thereby supporting worldwide economic growth.

On the flip side, when the FTSE 100 performs poorly, it can create negative ripple effects throughout the global economy. A downturn in the UK stock market can lower investor confidence and result in a reduction in investment and economic activity, both in the UK and in other nations closely tied to the UK economy.

In [38]:
```
ftse100_ticker = ["AAL.L", "ABF.L", "ADM.L", "ADN.L", "AGK.L", "AMEC.L",
                  "ANTO.L", "ARM.L", "ASHM.L", "AV.L", "AZN.L", "BA.L",
                  "BARC.L", "BATS.L", "BG.L", "BLND.L", "BLT.L", "BNZL.L"
                  "BP.L", "BRBY.L", "BSY.L", "BT-A.L", "CCL.L", "CNA.L",
                  "CPG.L", "CPI.L", "CRDA.L", "CRH.L", "CSCG.L", "DGE.L",
                  "EMG.L", "ENRC.L", "EVR.L", "EXPN.L", "FRES.L", "GFS.L"
                  "GKN.L", "GLEN.L", "GSK.L", "HL.L", "HMSO.L", "HSBA.L",
                  "IAG.L", "IAP.L", "IHG.L", "IMI.L", "IMT.L", "IPR.L",
                  "ITRK.L", "ITV.L", "JMAT.L", "KAZ.L", "KGF.L", "LAND.L"
                  "LGEN.L", "LLOY.L", "MGGT.L", "MKS.L", "MRW.L", "NG.L",
                  "NXT.L", "OML.L", "PFC.L", "POLY.L", "PRU.L", "PSON.L",
                  "RB.L", "RBS.L", "RDSB.L", "REL.L", "REX.L", "RIO.L",
                  "RR.L", "RRS.L", "RSA.L", "RSL.L", "SAB.L", "SBRY.L",
                  "SDR.L", "SDRC.L", "SGE.L", "SHP.L", "SL.L", "SMIN.L",
                  "SN.L", "SRP.L", "SSE.L", "STAN.L", "SVT.L", "TATE.L",
                  "TLW.L", "TSCO.L", "ULVR.L", "UU.L", "VED.L", "VOD.L",
                  "WEIR.L", "WOS.L", "WPP.L", "WTB.L", "XTA.L"]
```

In [39]:
```
uk_price_data = yf.download(ftse100_ticker, start="2023-03-13", end="2023
uk_price_data.dropna(axis=1, inplace=True)
```

```
[**********************100%************************]  101 of 101 completed

32 Failed downloads:
- GKN.L: No data found for this date range, symbol may be delisted
- IAP.L: No data found for this date range, symbol may be delisted
- VED.L: No data found for this date range, symbol may be delisted
- IMT.L: No data found for this date range, symbol may be delisted
- ENRC.L: No data found for this date range, symbol may be delisted
- RSL.L: No data found for this date range, symbol may be delisted
- SDRC.L: No timezone found, symbol may be delisted
- WOS.L: No data found for this date range, symbol may be delisted
- IPR.L: No data found for this date range, symbol may be delisted
- CSCG.L: No data found for this date range, symbol may be delisted
- AGK.L: No timezone found, symbol may be delisted
- ADN.L: No data found for this date range, symbol may be delisted
- XTA.L: No data found for this date range, symbol may be delisted
- RDSB.L: No timezone found, symbol may be delisted
- RSA.L: No timezone found, symbol may be delisted
- AMEC.L: No data found for this date range, symbol may be delisted
- BSY.L: No data found for this date range, symbol may be delisted
- SAB.L: No data found for this date range, symbol may be delisted
- SL.L: No data found for this date range, symbol may be delisted
- RRS.L: No data found for this date range, symbol may be delisted
- RB.L: No timezone found, symbol may be delisted
- REX.L: No data found for this date range, symbol may be delisted
- MGGT.L: No timezone found, symbol may be delisted
- BG.L: No data found for this date range, symbol may be delisted
- MRW.L: No timezone found, symbol may be delisted
- ARM.L: No data found for this date range, symbol may be delisted
- RBS.L: No timezone found, symbol may be delisted
- OML.L: No data found for this date range, symbol may be delisted
- KAZ.L: No timezone found, symbol may be delisted
- BLT.L: No data found for this date range, symbol may be delisted
- GFS.L: No timezone found, symbol may be delisted
- SHP.L: No data found for this date range, symbol may be delisted
```

In [40]:
```
df_weekly = pd.DataFrame(data=uk_price_data)
df_weekly.columns = df_weekly.columns.str.title()
df_weekly.tail(5)
```

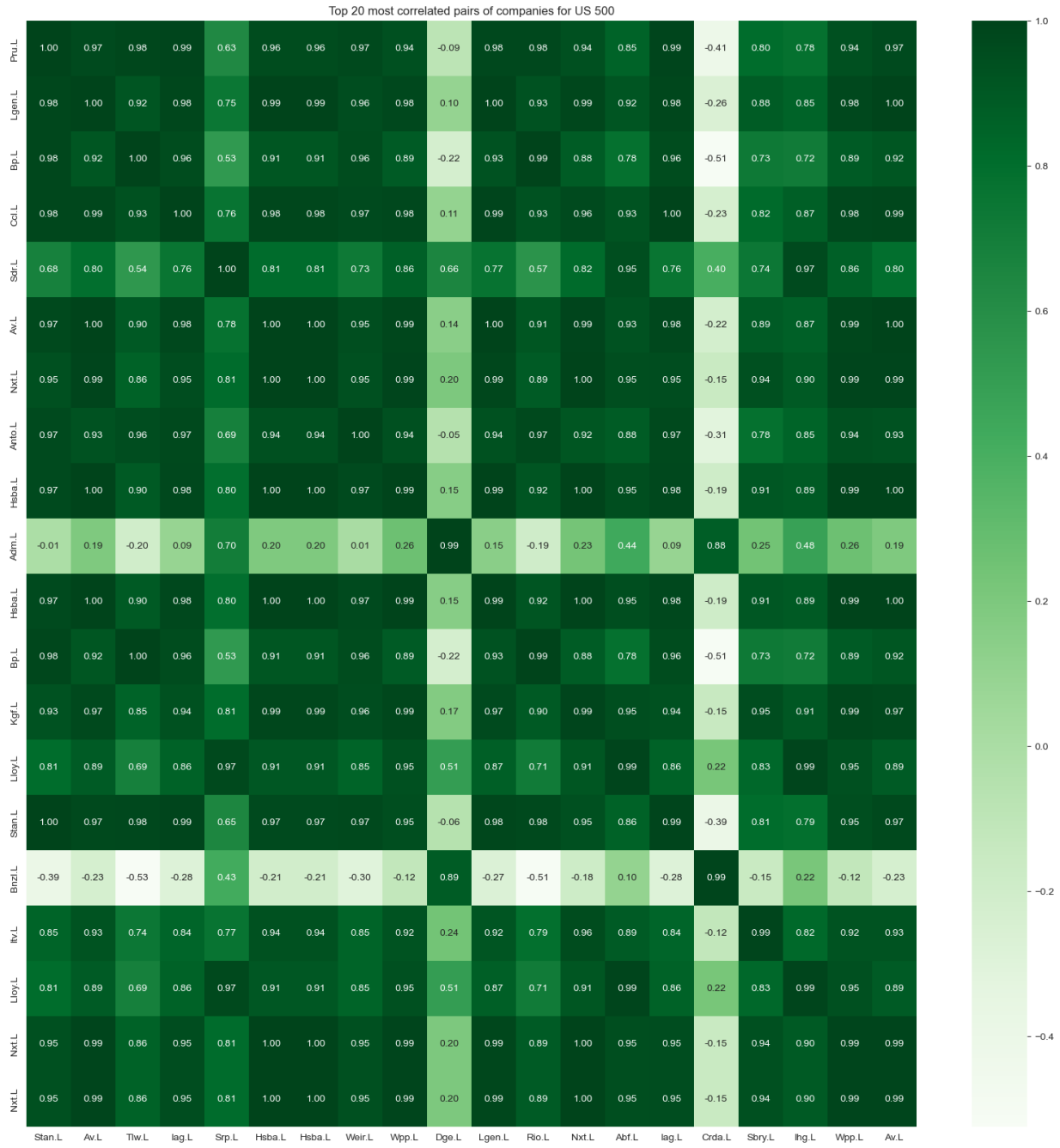
Out[40]:

Date	Aal.L	Abf.L	Adm.L	Anto.L	Ashm.L	Av.L	Azn.L	
2023-03-13 00:00:00	2649.128418	1948.0	1912.0	1500.5	235.800003	424.299988	10686.0	908.0
2023-03-14 00:00:00	2689.914062	1989.5	1920.0	1539.0	243.800003	428.600006	10686.0	938.0
2023-03-15 00:00:00	2485.500000	1900.0	1896.5	1445.5	233.800003	407.000000	10644.0	911.5
2023-03-16 00:00:00	2479.000000	1949.5	1954.5	1454.0	239.399994	415.000000	10798.0	916.7
2023-03-17 00:00:00	2504.000000	1901.5	1914.5	1444.0	231.199997	402.200012	10814.0	905.5

5 rows × 69 columns

In [41]:
```
# Compute the correlation matrix
corr_matrix = uk_price_data.corr()

# Find the top 20 most correlated pairs of companies
corr_pairs = corr_matrix.unstack().sort_values(ascending=False)
corr_pairs = corr_pairs[corr_pairs != 1].drop_duplicates()[:20]
corr_pairs = pd.DataFrame(corr_pairs).reset_index()
```

In [42]:
```
cmap = 'Greens'
sns.set_style("white")
plt.figure(figsize=(20, 20))
sns.heatmap(corr_matrix.loc[corr_pairs.iloc[:,0], corr_pairs.iloc[:,1]],
plt.title("Top 20 most correlated pairs of companies for US 500")
plt.show()
```



ANALYSIS OF SIX MONTHS DATA SET OF TOP UK 100 COMPANIES

In [43]:
```
uk_price_data = yf.download(ftse100_ticker, start="2022-09-21", end="2023
uk_price_data.dropna(axis=1, inplace=True)
```

```
[**********100%**********]  101 of 101 completed

32 Failed downloads:
- GKN.L: No data found for this date range, symbol may be delisted
- IAP.L: No data found for this date range, symbol may be delisted
- VED.L: No data found for this date range, symbol may be delisted
- IMT.L: No data found for this date range, symbol may be delisted
- ENRC.L: No data found for this date range, symbol may be delisted
- RSL.L: No data found for this date range, symbol may be delisted
- SDRC.L: No timezone found, symbol may be delisted
- WOS.L: No data found for this date range, symbol may be delisted
- IPR.L: No data found for this date range, symbol may be delisted
- CSCG.L: No data found for this date range, symbol may be delisted
- AGK.L: No timezone found, symbol may be delisted
- ADN.L: No data found for this date range, symbol may be delisted
- XTA.L: No data found for this date range, symbol may be delisted
- RDSB.L: No timezone found, symbol may be delisted
- RSA.L: No timezone found, symbol may be delisted
- AMEC.L: No data found for this date range, symbol may be delisted
- BSY.L: No data found for this date range, symbol may be delisted
- SAB.L: No data found for this date range, symbol may be delisted
- SL.L: No data found for this date range, symbol may be delisted
- RRS.L: No data found for this date range, symbol may be delisted
- RB.L: No timezone found, symbol may be delisted
- REX.L: No data found for this date range, symbol may be delisted
- MGGT.L: No timezone found, symbol may be delisted
- BG.L: No data found for this date range, symbol may be delisted
- MRW.L: No timezone found, symbol may be delisted
- ARM.L: No data found for this date range, symbol may be delisted
- RBS.L: No timezone found, symbol may be delisted
- OML.L: No data found for this date range, symbol may be delisted
- KAZ.L: No timezone found, symbol may be delisted
- BLT.L: No data found for this date range, symbol may be delisted
- GFS.L: No timezone found, symbol may be delisted
- SHP.L: No data found for this date range, symbol may be delisted
```

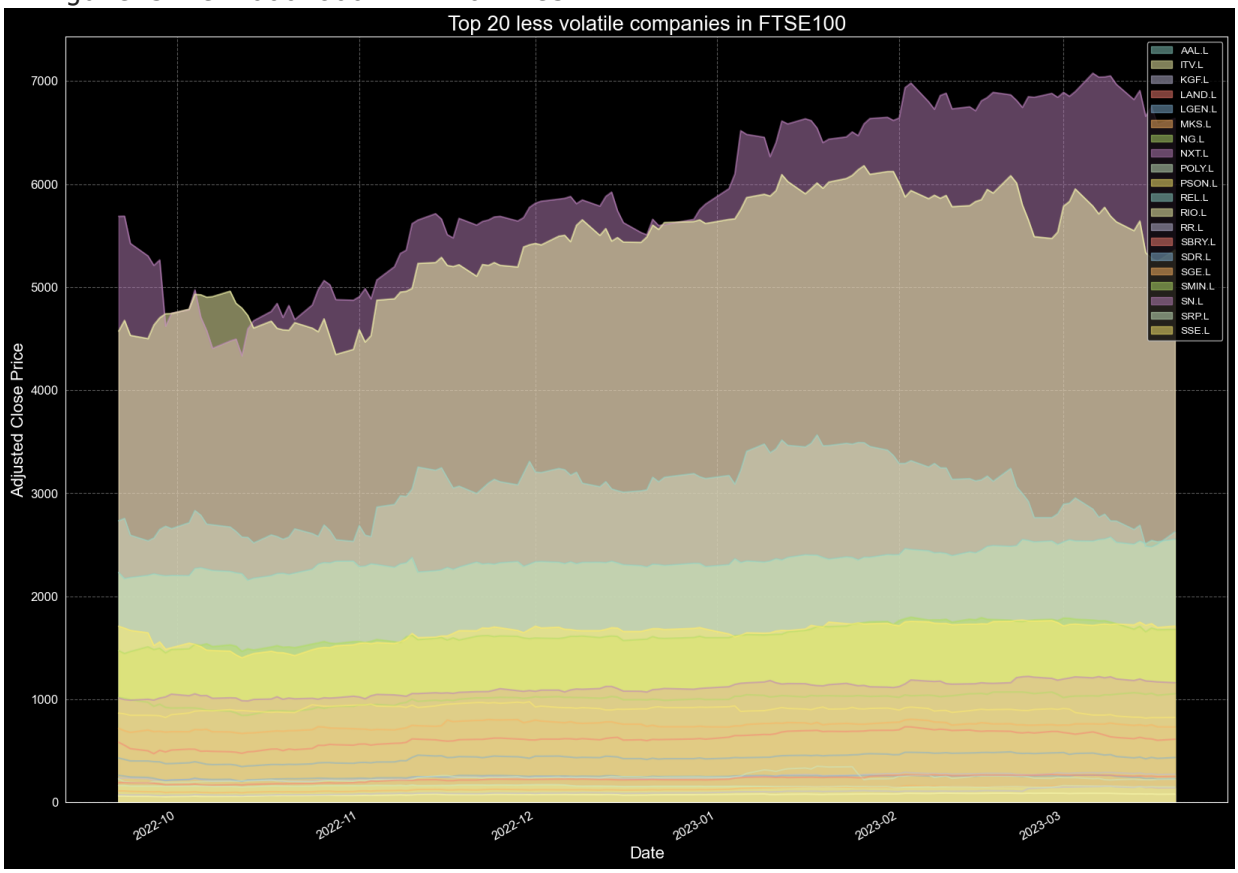
TOP 20 LESS VOLATILE COMPANIES IN FTSE100 (LAST 6 MONTHS)

In [44]:
```python
#less volatile companies based on technical analysis
rsi_data = pd.DataFrame()
for col in uk_price_data.columns:
    rsi = ta.momentum.RSIIndicator(uk_price_data[col]).rsi()
    rsi_data[col] = rsi

#companies with RSI < 30
low_rsi = rsi_data.iloc[-1] < 30
```

In [45]:
```python
#top 20 less volatile companies
safe_companies = low_rsi.sort_values().index[:20]
```

In [46]:
```
# Plot the prices of the top 20 safe companies as an area graph
plt.style.use('dark_background')
plt.figure(figsize=(20,6))
uk_price_data[safe_companies].plot(kind='area', alpha=0.5, stacked=False,
plt.title("Top 20 less volatile companies in FTSE100", fontsize=20)
plt.xlabel("Date", fontsize=16)
plt.ylabel("Adjusted Close Price", fontsize=16)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.grid(linestyle='--', color='gray', alpha=0.7)
plt.show()
plt.close()
```

```
<Figure size 2000x600 with 0 Axes>
```

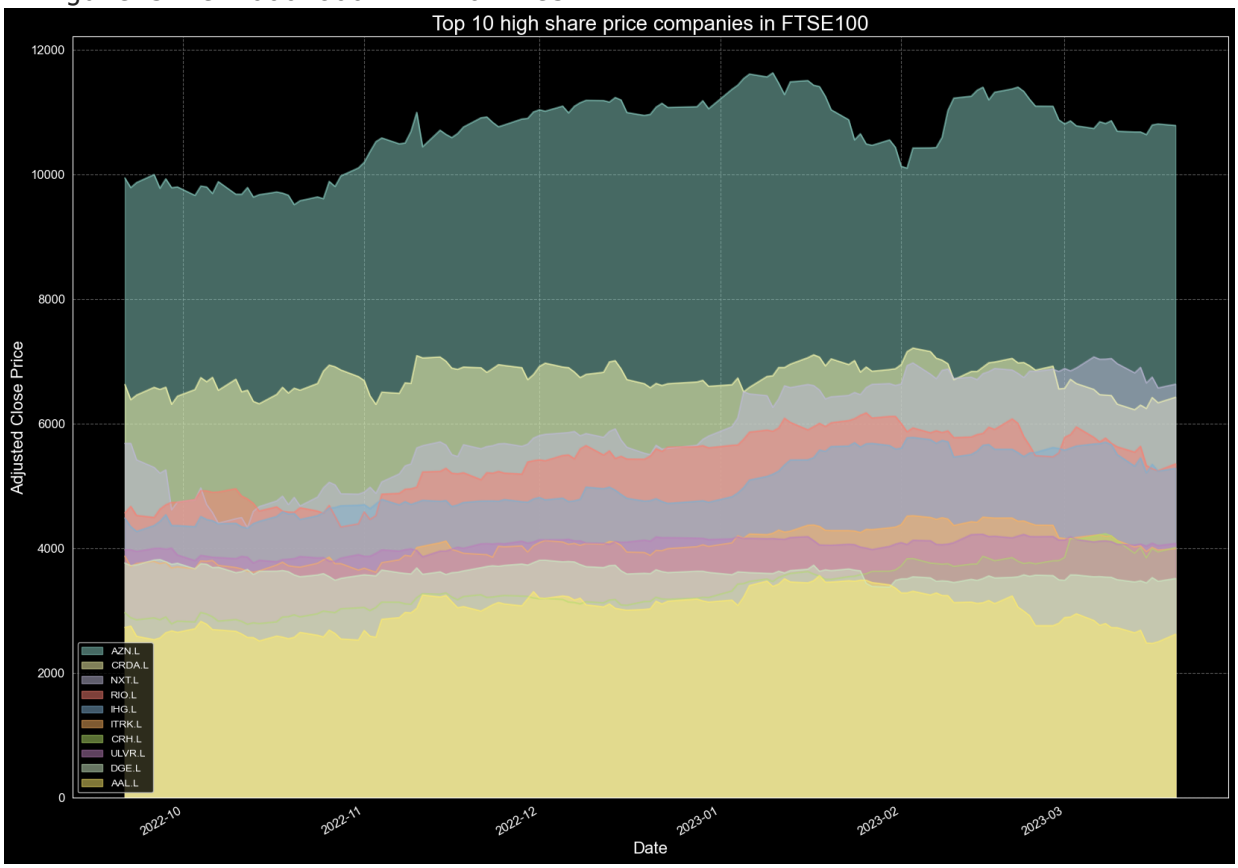


TOP 10 HIGH SHARE PRICE AND LOW SHARE PRICE IN FTSE100 (IAST SIX MONTHS)

In [47]:
```
high_price_companies = uk_price_data.max().sort_values(ascending=False).i
low_price_companies = uk_price_data.max().sort_values().index[:10]
```

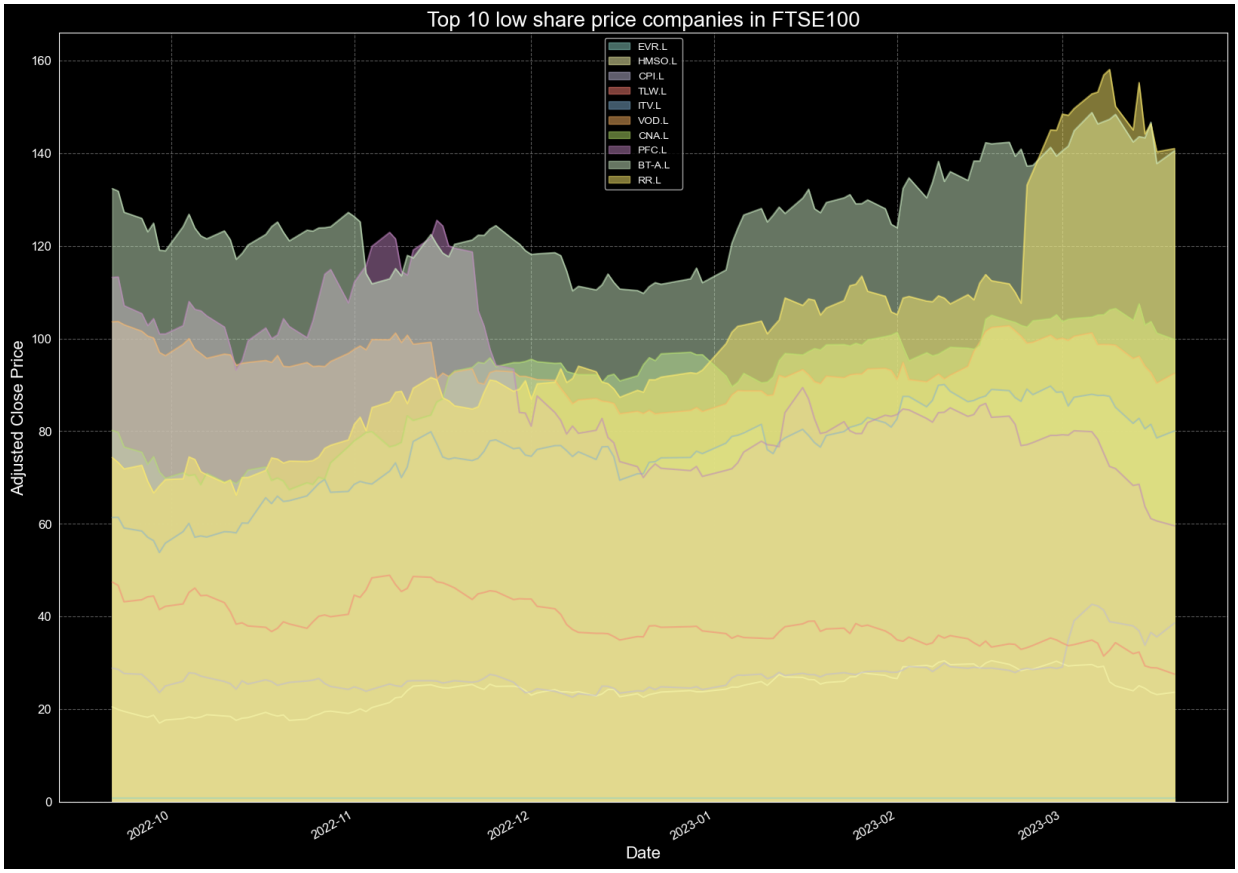
In [48]:
```
#top high share price companies as an area graph
plt.style.use('dark_background')
plt.figure(figsize=(20,6))
uk_price_data[high_price_companies].plot(kind='area', alpha=0.5, stacked=
plt.title("Top 10 high share price companies in FTSE100", fontsize=20)
plt.xlabel("Date", fontsize=16)
plt.ylabel("Adjusted Close Price", fontsize=16)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.grid(linestyle='--', color='gray', alpha=0.7)
plt.show()
plt.close()
```

```
<Figure size 2000x600 with 0 Axes>
```

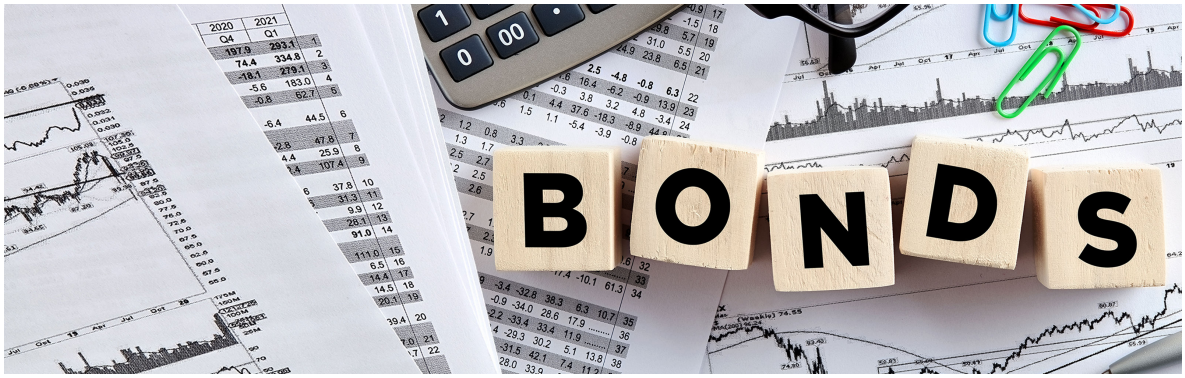


In [49]:
```
# Plot the prices of top low share price companies as an area graph
plt.style.use('dark_background')
plt.figure(figsize=(20,6))
uk_price_data[low_price_companies].plot(kind='area', alpha=0.5, stacked=F
plt.title("Top 10 low share price companies in FTSE100", fontsize=20)
plt.xlabel("Date", fontsize=16)
plt.ylabel("Adjusted Close Price", fontsize=16)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.grid(linestyle='--', color='gray', alpha=0.7)
plt.show()
plt.close()
```

```
<Figure size 2000x600 with 0 Axes>
```



4. BONDS



Bonds are debt securities that are issued by governments, corporations, and other entities to raise capital. They are essentially loans that investors make to the issuer in exchange for periodic interest payments and the return of the face value of the bond when it matures.

1. Government bonds: Bonds issued by national, state, or local governments.
2. Corporate bonds: Bonds issued by corporations to raise capital for various purposes, such as funding expansion plans or repaying existing debt.
3. Municipal bonds: Bonds issued by cities, counties, or other local government entities to finance public projects or services.
4. Treasury bonds: Bonds issued by the federal government to finance the national debt and support the economy.

5. Agency bonds: Bonds issued by government-sponsored entities, such as Freddie Mac and Fannie Mae.
6. High-yield bonds: Also known as junk bonds, these are bonds issued by companies with lower credit ratings and therefore higher risk of default. They offer higher yields to compensate for this risk.
7. Floating-rate bonds: Bonds with a variable interest rate that changes periodically based on a benchmark rate, such as the London Interbank Offered Rate (LIBOR).
8. Convertible bonds: Bonds that can be converted into a specified number of shares of the issuer's common stock.
9. Zero-coupon bonds: Bonds that are sold at a discount to face value and pay no periodic interest, but instead pay the full face value when they mature.
10. Asset-backed bonds: Bonds backed by a specific asset, such as a mortgage or consumer loan.
11. Floating rate notes (FRNs): Similar to floating-rate bonds, but usually issued by financial institutions.
12. Exchange-traded bonds (ETBs): Bonds that are traded on an exchange, like stocks.
13. Foreign bonds: Bonds issued by a foreign entity, such as a foreign government or corporation.
14. Perpetual bonds: Bonds with no maturity date, meaning they pay interest indefinitely.
15. Callable bonds: Bonds that can be redeemed by the issuer before the maturity date.
16. Puttable bonds: Bonds that can be sold back to the issuer before the maturity date.

bonds can be seen as a hedge against inflation, as the interest paid on bonds usually increases with inflation. This means that as the cost of goods and services increases, the return on bonds increases as well, helping to offset some of the negative effects of inflation.

However, during periods of high inflation, the value of bonds can decline due to the increased cost of borrowing and declining purchasing power of the currency. This can result in losses for bondholders.

During a recession, the demand for bonds usually increases as investors seek safe, low-risk investments. This can lead to lower interest rates and an increase in bond prices. However, if the recession is severe and long-lasting, it can lead to a decline in the creditworthiness of many companies, resulting in declining bond prices and increased risk of default.

5. TECHNICAL ANALYSIS

1. Long-term investments: A long-term investment is typically an investment with a holding period of more than one year. When considering a long-term investment in a company, it is important to conduct a fundamental analysis of the company at least once a year. This will provide an updated view of the company's financial performance, stability, and growth prospects. Based on this information, you can adjust your investment strategy accordingly to ensure your investment aligns with your long-term goals.

2. Short-term investments: A short-term investment is typically an investment with a holding period of less than one year. When considering a short-term investment in a company, it is important to conduct a fundamental analysis more frequently, such as every quarter or every six months. This will provide an updated view of the company's financial performance and stability, which can change quickly in the short term. Based on this information, you can adjust your investment strategy accordingly to ensure your investment aligns with your short-term goals.

3. High-risk investments: High-risk investments are investments that carry a higher level of risk, such as stocks in small, unproven companies or stocks in industries that are facing challenges. When considering a high-risk investment, it is important to conduct a fundamental analysis more frequently to stay informed about the financial performance and stability of the company. This will help you make informed decisions about the investment and minimize the risks associated with the investment.

4. Low-risk investments: Low-risk investments are investments that carry a lower level of risk, such as bonds or blue-chip stocks. When considering a low-risk investment, it may not be necessary to conduct a fundamental analysis as frequently as you would for a high-risk investment. However, it is still important to review the investment regularly and conduct a fundamental analysis as needed to ensure your investment aligns with your goals.

1. Total Period of Trading (Maximum Chart)

In [50]:
```
ticker = "CS"
stock = yf.download(ticker)
stock.shape
```

```
[**********100%**********]  1 of 1 completed
```

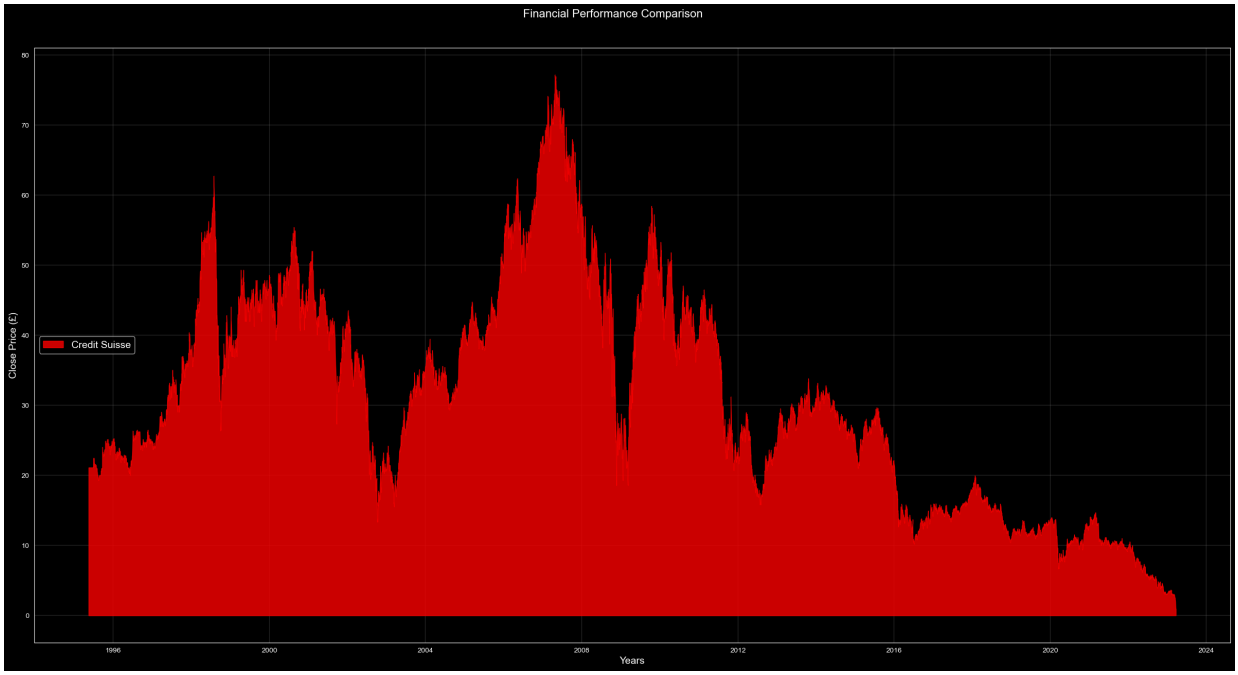
Out[50]:
```
(7012, 6)
```

In [51]:
```
stock.tail()
```

Out[51]:

Date	Open	High	Low	Close	Adj Close	Volume
2023-03-16	2.30	2.32	2.11	2.16	2.16	224765200
2023-03-17	2.05	2.10	1.94	2.01	2.01	136229600
2023-03-20	0.91	1.08	0.90	0.94	0.94	375645900
2023-03-21	0.97	1.00	0.94	0.97	0.97	210272500
2023-03-22	0.94	0.96	0.91	0.92	0.92	100043800

In [52]:
```
plt.style.use('dark_background')
plt.figure(figsize=(30,15))
plt.grid(True, color='grey', linewidth=0.3)
plt.fill_between(stock.index, 0, stock['Close'], color="red", label='Cred
plt.xlabel('Years', fontsize=14)
plt.ylabel('Close Price (£)', fontsize=14)
plt.legend(loc='center left', fontsize=14)
plt.suptitle("Financial Performance Comparison", fontsize=16, y=0.93)
plt.show()
```



In [53]:
```
plt.style.use('dark_background')
plt.figure(figsize=(30,15))
plt.plot(stock['Open'], color='tab:green', label='Open Price')
plt.plot(stock['Close'], color='tab:red', label='Close Price')
plt.plot(stock['Adj Close'], color='tab:blue', label='Adj Close Price')
plt.title("Credit Suisse Group AG (CS) Stock Price", fontsize=18)
plt.xlabel("Year", fontsize=14)
plt.ylabel("Price (USD)", fontsize=14)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.legend(fontsize=14)
plt.grid(linestyle='--', color='gray', alpha=0.5)
plt.show()
```

1. ThresholdClassifier Algortihms Design by Hemant Thapa

This classifier will work with respect to time series analysis, it can be varies from 1months, 6months, 1years, 5years, 10years, and Total trading years. The algorithm takes in a list of numerical data, calculates the mean value of the data, and sets the threshold value to be equal to the mean. The algorithm then iterates through each value in the data list and compares it to the threshold value. If the value is greater than the threshold value, the algorithm classifies it as a 1 (positive class), otherwise, it classifies it as a 0 (negative class).

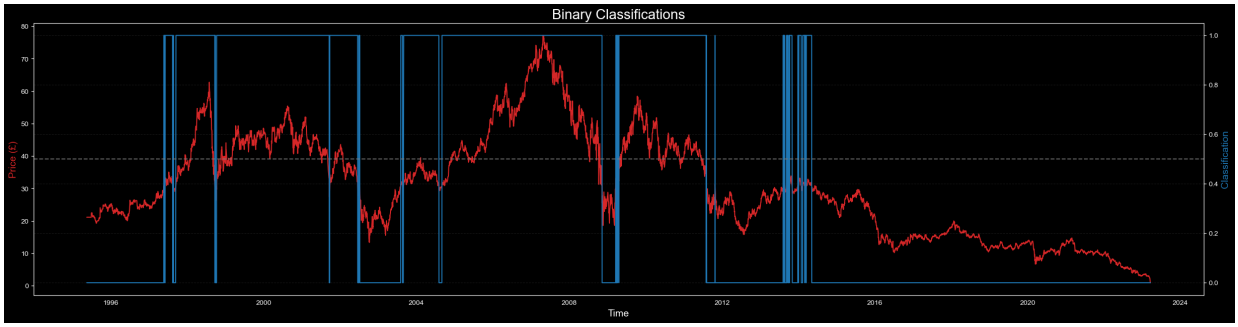
It will help to you identifies the positive and negative trade over your trading period.

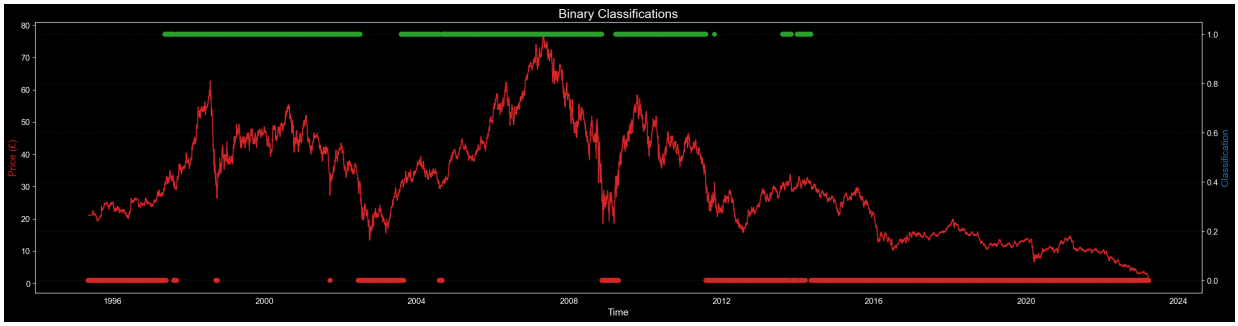
In [59]:
```
class ThresholdClassifier:
    def __init__(self, threshold: float):
        self.threshold = threshold

    def classify(self, data: List[float]):
        self.mean = sum(data) / len(data)
        self.threshold = self.mean
        classifications = []
        for value in data:
            if value > self.threshold:
                classifications.append(1)
            else:
                classifications.append(0)
        return classifications
```

In [60]:
```
classifier = ThresholdClassifier(0.5)
```

In [61]:
```
class ThresholdPlot:
    def __init__(self, data, classification):
        self.data = data
        self.classification = classification

    def plot(self):
        plt.style.use('dark_background')
        plt.figure(figsize=(30,7))
        plt.xlabel("Time", fontsize=14)
        plt.ylabel("Price (£)", color="tab:red", fontsize=14)
        plt.plot(self.data.index, self.data, color="tab:red")
        plt.twinx()
        plt.ylabel("Classification", color="tab:blue", fontsize=14)
        plt.plot(self.data.index, self.classification, color="tab:blue")
        plt.axhline(y=0.5, color="gray", linestyle="--", alpha=0.8)
        plt.title("Binary Classifications", fontsize=20, color="White")
        plt.grid(linestyle='--', color='gray', alpha=0.2)
        plt.show()

        colors = []
        for value in self.classification:
            if value > 0.5:
                colors.append("tab:green")
            else:
                colors.append("tab:red")

        plt.style.use('dark_background')
        plt.figure(figsize=(30,7))
        plt.xlabel("Time", fontsize=14)
        plt.ylabel("Price (£)", color="tab:red", fontsize=14)
        plt.plot(self.data.index, self.data, color="tab:red")
        plt.tick_params(axis="both", labelsize=12)
        plt.twinx()
        plt.ylabel("Classification", color="tab:blue", fontsize=14)
        plt.scatter(self.data.index, self.classification, color=colors)
        plt.tick_params(axis="both", labelsize=12)
        plt.title("Binary Classifications", fontsize=18, color="White")
        plt.grid(linestyle='--', color='gray', alpha=0.2)
        plt.show()
```

In [62]:
```
classifier = ThresholdClassifier(0.5)
classification = classifier.classify(stock['Close'].tolist())
plot = ThresholdPlot(stock['Close'], classification)
plot.plot()
```





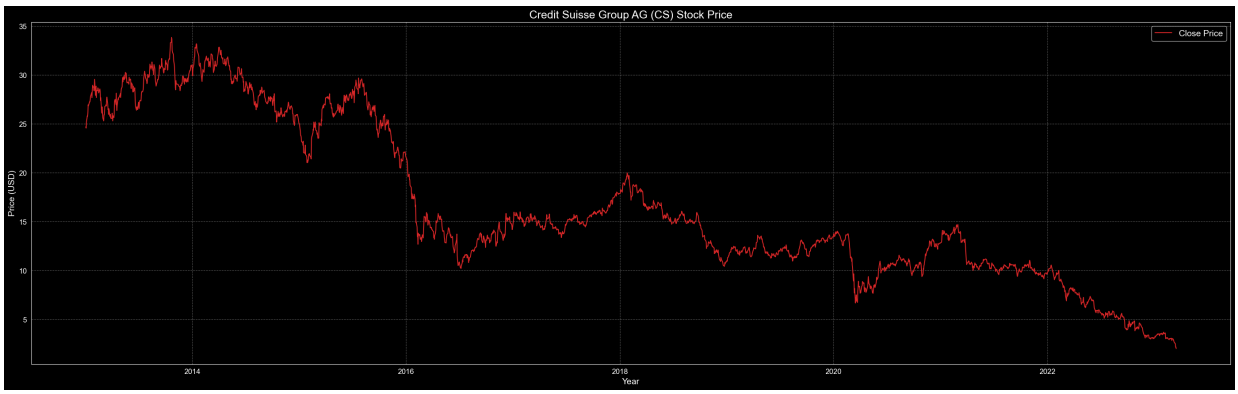
10 YEAR CHART

In [63]:
```
ticker = "CS"
stock = yf.download(ticker, start="2013-01-01", end="2023-03-20")
stock.shape
```

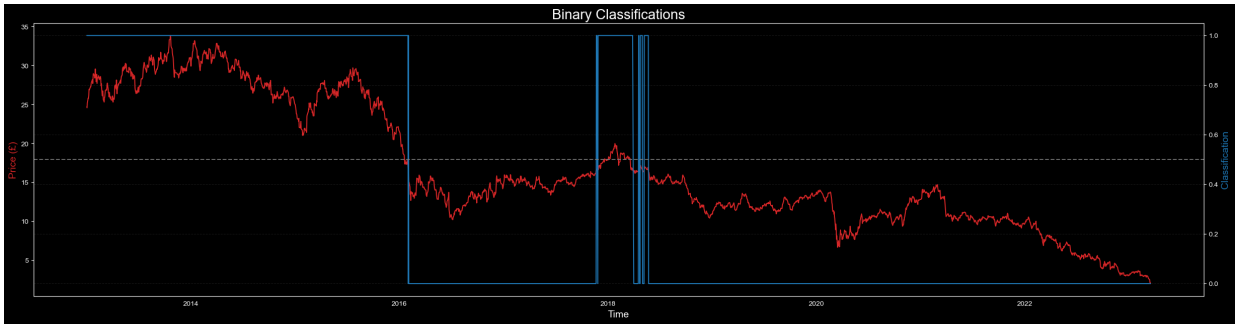
```
[**********100%**********]  1 of 1 completed
```

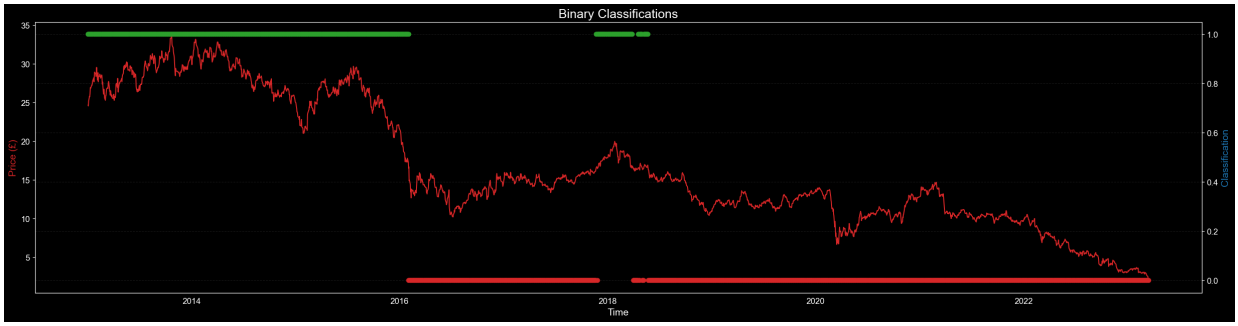
Out[63]:
```
(2570, 6)
```

In [64]:
```
plt.style.use('dark_background')
plt.figure(figsize=(35,10))
plt.plot(stock['Close'], color='tab:red', label='Close Price')
plt.title("Credit Suisse Group AG (CS) Stock Price", fontsize=18)
plt.xlabel("Year", fontsize=14)
plt.ylabel("Price (USD)", fontsize=14)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.legend(fontsize=14)
plt.grid(linestyle='--', color='gray', alpha=0.7)
plt.show()
```



In [65]:
```
classifier = ThresholdClassifier(0.5)
classification = classifier.classify(stock['Close'].tolist())
plot = ThresholdPlot(stock['Close'], classification)
plot.plot()
```





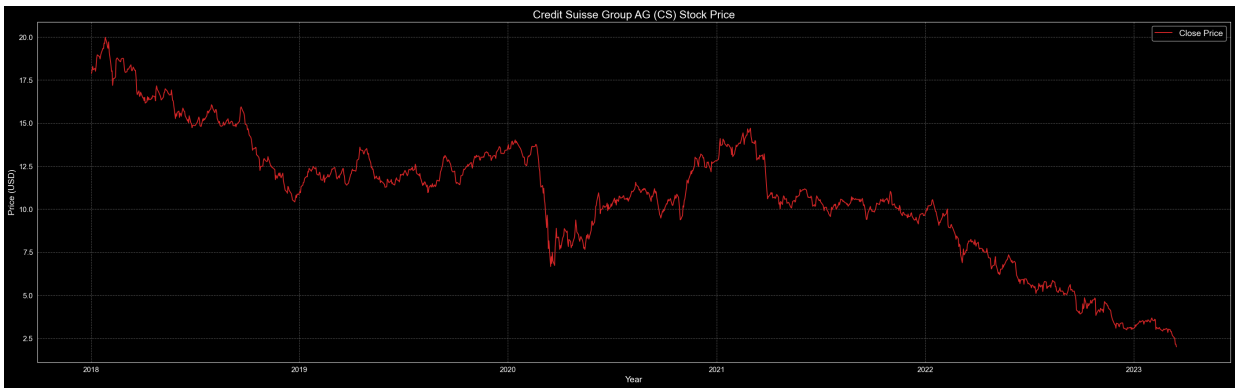
5 YEAR CHART

In [66]:
```
ticker = "CS"
stock = yf.download(ticker, start="2018-01-01", end="2023-03-20")
stock.shape
```

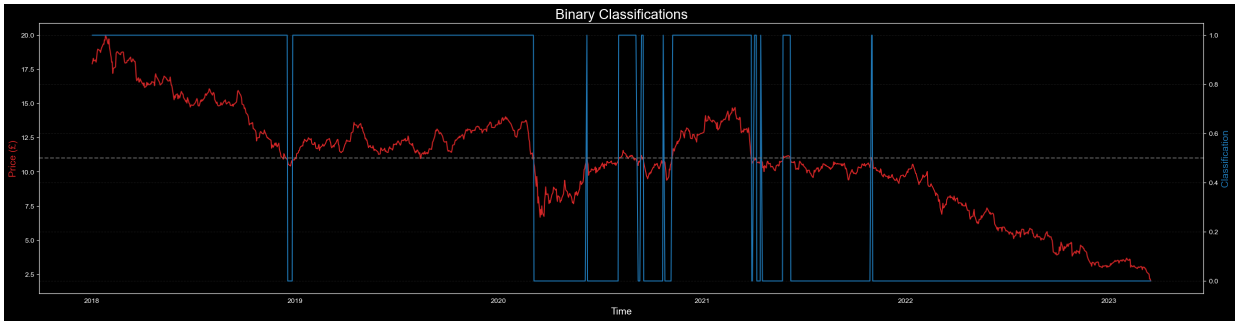
```
[**********100%**********]  1 of 1 completed
```

Out[66]:
```
(1311, 6)
```

In [67]:
```
plt.style.use('dark_background')
plt.figure(figsize=(35,10))
plt.plot(stock['Close'], color='tab:red', label='Close Price')
plt.title("Credit Suisse Group AG (CS) Stock Price", fontsize=18)
plt.xlabel("Year", fontsize=14)
plt.ylabel("Price (USD)", fontsize=14)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.legend(fontsize=14)
plt.grid(linestyle='--', color='gray', alpha=0.7)
plt.show()
```



In [68]:
```
classifier = ThresholdClassifier(0.5)
classification = classifier.classify(stock['Close'].tolist())
plot = ThresholdPlot(stock['Close'], classification)
plot.plot()
```



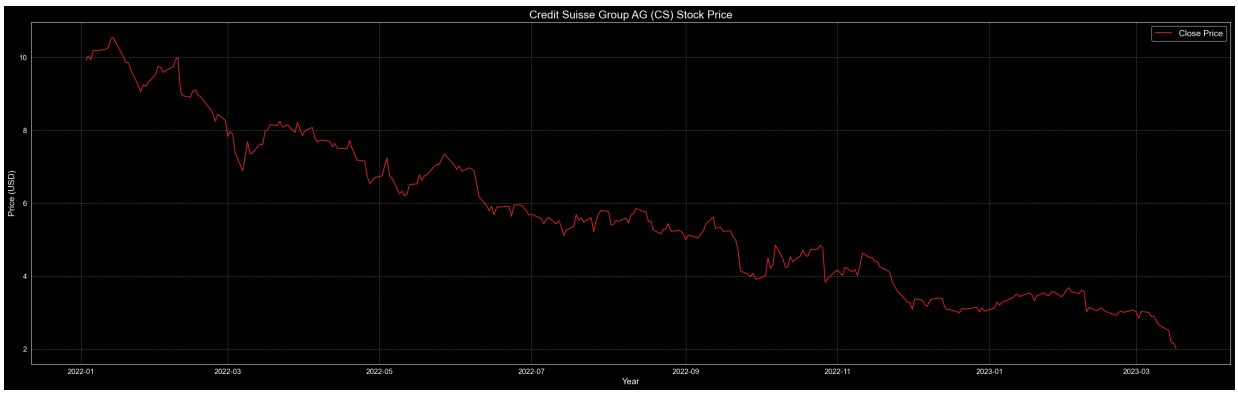
1 YEAR CHART

In [69]:
```
ticker = "CS"
stock = yf.download(ticker, start="2022-01-01", end="2023-03-20")
stock.shape
```

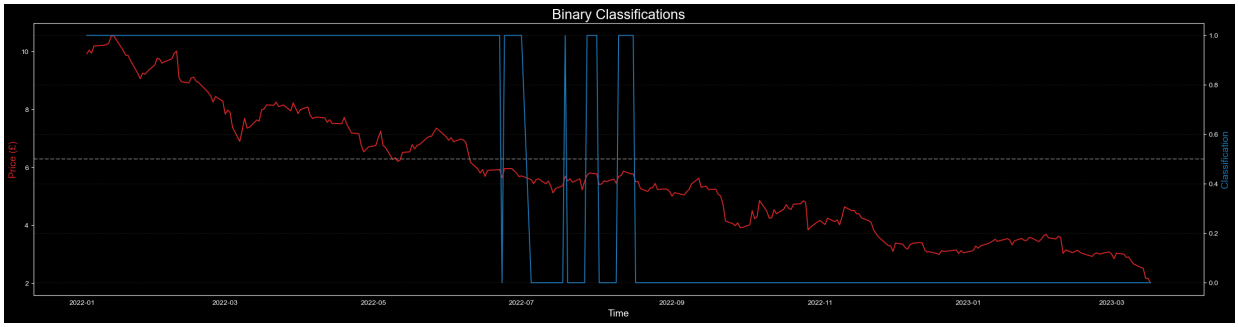
```
[**********100%**********]  1 of 1 completed
```

Out[69]:
```
(303, 6)
```

In [70]:
```
plt.style.use('dark_background')
plt.figure(figsize=(35,10))
plt.plot(stock['Close'], color='tab:red', label='Close Price')
plt.title("Credit Suisse Group AG (CS) Stock Price", fontsize=18)
plt.xlabel("Year", fontsize=14)
plt.ylabel("Price (USD)", fontsize=14)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.legend(fontsize=14)
plt.grid(linestyle='--', color='gray', alpha=0.7)
plt.show()
```



In [71]:
```
classifier = ThresholdClassifier(0.5)
classification = classifier.classify(stock['Close'].tolist())
plot = ThresholdPlot(stock['Close'], classification)
plot.plot()
```



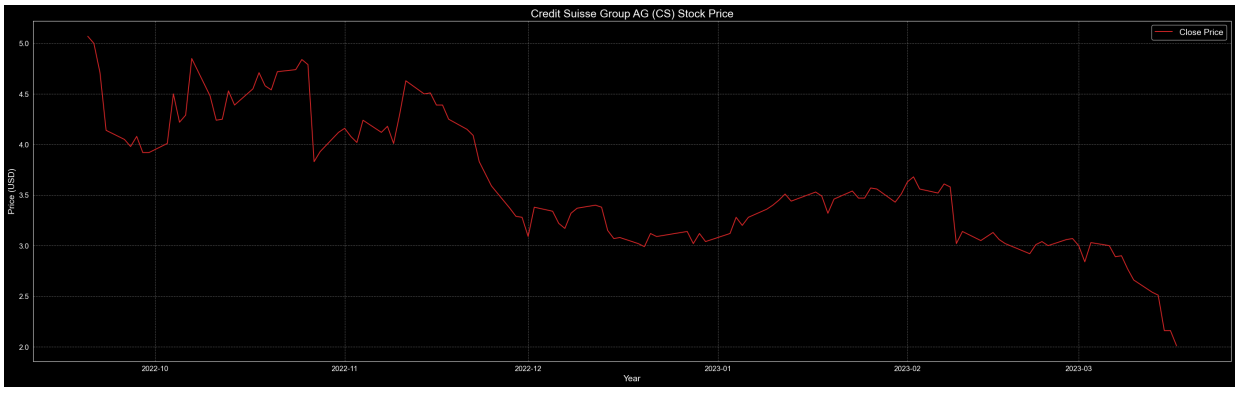
6 MONTHS CHART

In [72]:
```
ticker = "CS"
stock = yf.download(ticker, start="2022-09-20", end="2023-03-20")
stock.shape
```

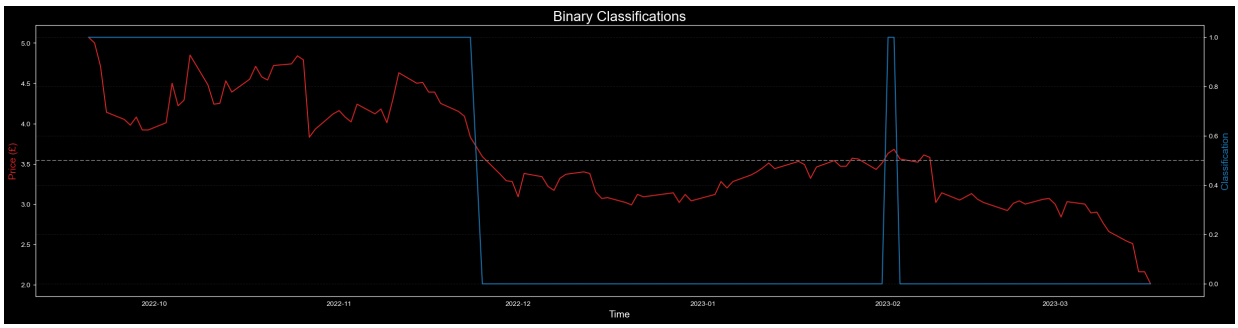
```
[**********100%**********]  1 of 1 completed
```

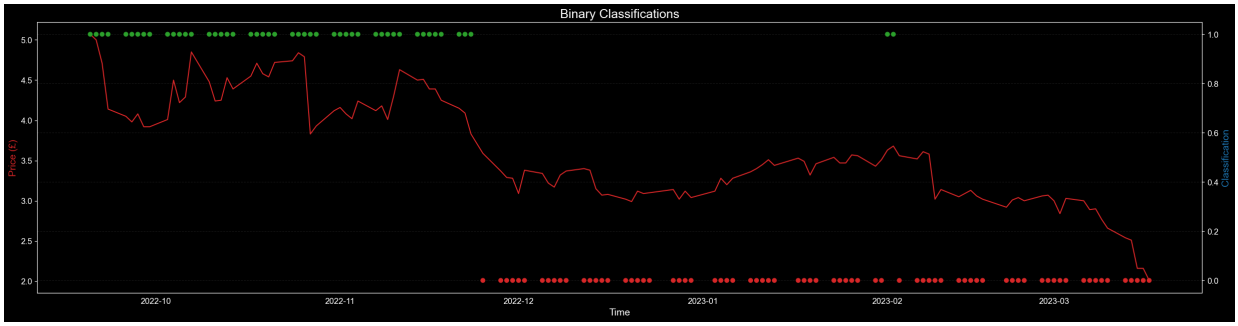
Out[72]:
```
(124, 6)
```

In [73]:
```
plt.style.use('dark_background')
plt.figure(figsize=(35,10))
plt.plot(stock['Close'], color='tab:red', label='Close Price')
plt.title("Credit Suisse Group AG (CS) Stock Price", fontsize=18)
plt.xlabel("Year", fontsize=14)
plt.ylabel("Price (USD)", fontsize=14)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.legend(fontsize=14)
plt.grid(linestyle='--', color='gray', alpha=0.7)
plt.show()
```



In [74]:
```
classifier = ThresholdClassifier(0.5)
classification = classifier.classify(stock['Close'].tolist())
plot = ThresholdPlot(stock['Close'], classification)
plot.plot()
```





2. LINEAR REGRESSION MODEL

Credit Suisse is one of the largest financial institutions in the world, and as such, its performance can have ripple effects on the global economy. Credit Suisse provides a range of financial services, including investment banking, wealth management, and asset management, and its operations span across multiple countries and regions.

The performance of Credit Suisse can affect the global economy in several ways. For example, if Credit Suisse performs well, it can attract investment and stimulate economic growth in the countries and regions where it operates. Conversely, if Credit Suisse experiences financial difficulties, it can lead to a reduction in investment and economic activity, which can have negative effects on the global economy.

Moreover, the collapse or failure of a major financial institution like Credit Suisse can have systemic effects on the global economy, particularly if it triggers a broader financial crisis. The failure of Lehman Brothers in 2008, for example, had far-reaching impacts on the global economy and was a major contributor to the global financial crisis that followed.

Credit Suisse's impact on the S&P 500 may be limited as Credit Suisse is a Swiss financial institution and is not included in the S&P 500 index. However, as a major global financial institution, the performance of Credit Suisse can still have indirect effects on the US economy and, by extension, the S&P 500.

Credit Suisse experiences financial difficulties, it may reduce its investment in the US and may even reduce its lending to US companies and individuals, which can affect the US economy and the performance of the S&P 500. Additionally, if Credit Suisse's financial difficulties are severe enough, it could contribute to broader economic trends and developments that could indirectly affect the S&P 500.

MAX CHART - CREDIT SUISSE AND SNP500

In [78]:
```
credit_suisse = yf.Ticker("CS").history(period="max")
snp500 = yf.Ticker("^GSPC").history(period="max")
ftse100 = yf.Ticker("^FTSE").history(period="max")
```

In [79]:
```
from forex_python.converter import CurrencyRates
```

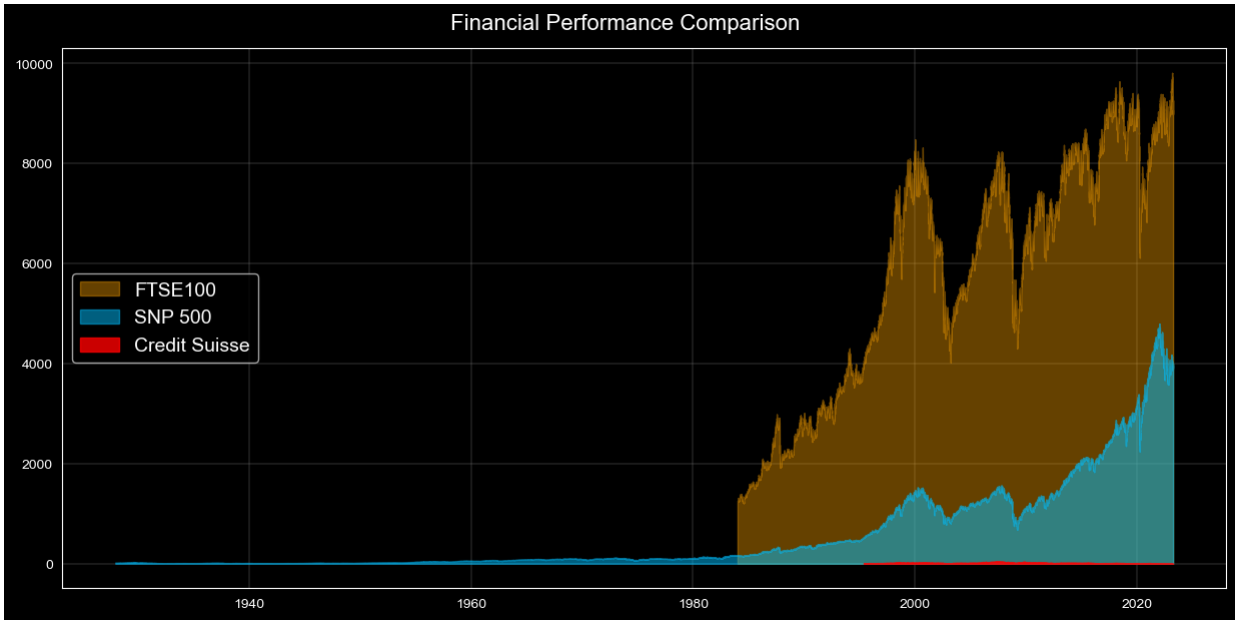
In [80]:
```
c = CurrencyRates()
exchange_rate = c.get_rate('GBP', 'USD')
ftse100["Close"] = ftse100["Close"] * exchange_rate
ftse100["Open"] = ftse100["Open"] * exchange_rate
ftse100["High"] = ftse100["High"] * exchange_rate
ftse100["Low"] = ftse100["Low"] * exchange_rate
```

In [81]:
```
ftse100.tail(2)
```

Out[81]:

Date	Open	High	Low	Close	Volume	Divi
2023-03-22 00:00:00+00:00	9224.960334	9285.430083	9187.380641	9262.416900	892439300	
2023-03-23 00:00:00+00:00	9262.465911	9262.465911	9151.429080	9164.967547	0	

In [82]:
```
plt.style.use('dark_background')
plt.figure(figsize=(15,7))
plt.grid(True, color='grey', linewidth=0.3)
plt.plot(snp500.index, snp500['Close'], alpha=0.2, color="#00bfff", linew
plt.fill_between(ftse100.index, 0, ftse100['Close'], color="orange", labe
plt.fill_between(snp500.index, 0, snp500['Close'], color="#00bfff", label
plt.fill_between(credit_suisse.index, 0, credit_suisse['Close'], color="r
plt.legend(loc='center left', fontsize=14)
plt.suptitle("Financial Performance Comparison", fontsize=16, y=0.93)
plt.show()
```



SNP500 is the oldest index in Global finance, The S&P 500 is not a stock itself, but rather a stock market index that measures the performance of the 500 largest publicly traded companies in the United States. The index was first introduced in 1923 by Standard & Poor's Financial Services and has undergone several revisions since then. first S&P 500 index fund was introduced by Vanguard in 1976.

The FTSE 100 Index was launched on January 3, 1984, by the Financial Times Stock Exchange. well, Credit Suisse Group AG, a Swiss multinational investment bank, was first established in 1856, but it did not go public until 1997. It was listed on the Zurich and New York stock exchanges on October 31, 1997.

5 YEAR CHART

A 5-year chart can also provide a more accurate representation of a company's financial performance over time. Short-term fluctuations and market noise may be smoothed out, giving a clearer picture of the company's growth and stability.

Credit Suisse

In [83]:
```
ticker = "CS"
stock = yf.download(ticker, start="2018-03-20", end="2023-03-20")
stock.shape
```

```
[**********100%**********]  1 of 1 completed
```

Out[83]:
```
(1258, 6)
```

In [84]:
```
plt.style.use('dark_background')
plt.figure(figsize=(35,10))
plt.plot(stock.index, stock['Close'], label='Credit Suisse', color="red")
plt.fill_between(stock.index, stock['Close'], 0, alpha=0.3, color='royalb
plt.title('SNP500 Stock Price', fontsize=20, color="White")
plt.xlabel("Time", fontsize=14)
plt.ylabel("Price ($)", fontsize=14)
plt.legend(loc='best')
plt.tick_params(axis="both", labelsize=12)
plt.grid(linestyle='--', color='gray', alpha=0.7)
plt.show()
```

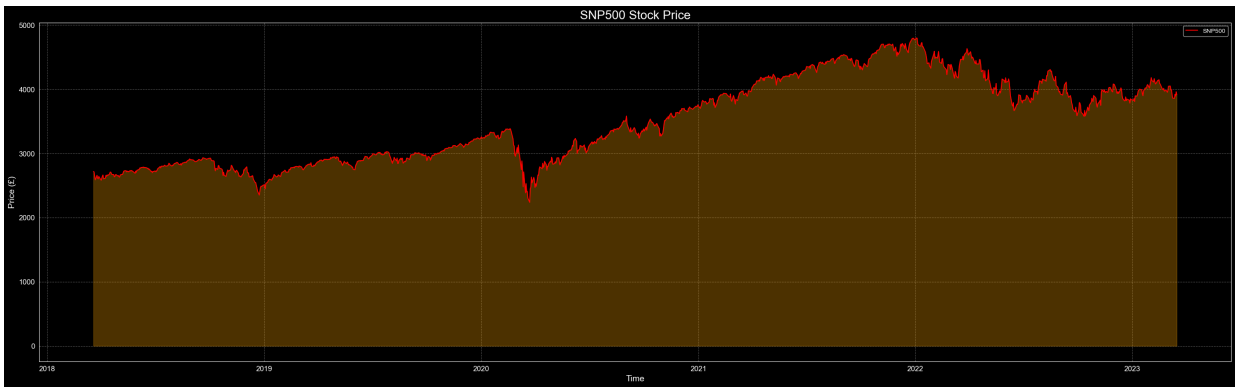
SNP500

In [85]:
```
ticker2 = "^GSPC"
stock2 = yf.download(ticker2, start="2018-03-20", end="2023-03-20")
stock2.shape
```

```
[**********100%**********]  1 of 1 completed
```

Out[85]:
```
(1258, 6)
```

In [86]:
```
plt.style.use('dark_background')
plt.figure(figsize=(35,10))
plt.plot(stock2.index, stock2['Close'], label='SNP500', color="red")
plt.fill_between(stock2.index, stock2['Close'], 0, alpha=0.3,color='orang
plt.title('SNP500 Stock Price', fontsize=20, color="White")
plt.xlabel("Time", fontsize=14)
plt.ylabel("Price (£)", fontsize=14)
plt.legend(loc='best')
plt.tick_params(axis="both", labelsize=12)
plt.grid(linestyle='--', color='gray', alpha=0.7)
plt.show()
```



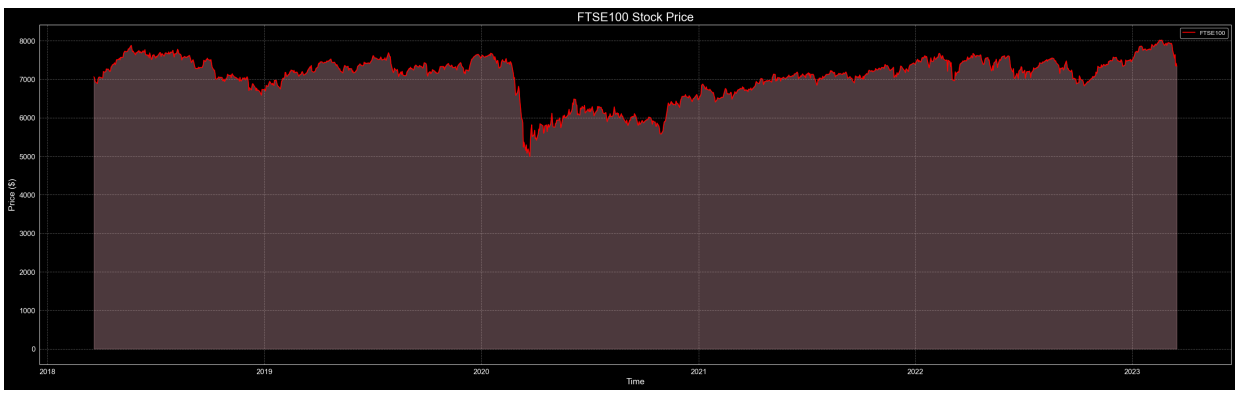
FTSE100

In [87]:
```
ticker3 = "^FTSE"
stock3 = yf.download(ticker3, start="2018-03-20", end="2023-03-20")
stock3.shape
```

```
[**********100%**********]  1 of 1 completed
```

Out[87]:
```
(1261, 6)
```

In [88]:
```
plt.style.use('dark_background')
plt.figure(figsize=(35,10))
plt.plot(stock3.index, stock3['Close'], label='FTSE100', color="red")
plt.fill_between(stock3.index, stock3['Close'], 0, alpha=0.3,color='pink'
plt.title('FTSE100 Stock Price', fontsize=20, color="White")
plt.xlabel("Time", fontsize=14)
plt.ylabel("Price ($)", fontsize=14)
plt.legend(loc='best')
plt.tick_params(axis="both", labelsize=12)
plt.grid(linestyle='--', color='gray', alpha=0.7)
plt.show()
```



SKLEARN LINEAR REGRESSION

In [39]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from datetime import datetime
```

PRICE ANALYSIS

TEN YEAR TIME SERIES ANALYSIS

In [40]:
```
cs = yf.Ticker("CS")
data = cs.history(period="10y")
```

In [41]:
```
data['Date_ordinal'] = [datetime.toordinal(d) for d in data.index]
X_train, X_test, y_train, y_test = train_test_split(data['Date_ordinal'].
```

In [42]:
```
# Create a linear regression model
model = LinearRegression()
```

In [43]:
```
# Train the model
model.fit(X_train, y_train)
```

Out[43]:
```
▾ LinearRegression

LinearRegression()
```

In [44]:
```
# Predict the values
y_pred = model.predict(X_test)
```

In [45]:
```
# Convert ordinal value back to date
X_test = [datetime.fromordinal(int(d)) for d in X_test.flatten()]
```

In [46]:
```
plt.style.use('dark_background')
plt.figure(figsize=(35,15))
sns.scatterplot(x=X_test, y=y_test, label='Actual', color='blue')
plt.plot(X_test, y_pred, label='Predicted', color='red')
plt.title('Linear Regression Model - 10 year chart', fontsize=30, color="
plt.xlabel("SNP500 ($)", fontsize=20, fontweight='bold')
plt.ylabel("Credit Suisse ($)", fontsize=20, fontweight='bold')
plt.tick_params(axis="both", labelsize=16)
plt.grid(linestyle='--', color='gray', alpha=0.7)
plt.show()
```

FIVE YEAR TIME SERIES ANALYSIS

In [47]:
```
cs = yf.Ticker("CS")
data = cs.history(period="5y")
```

In [48]:
```
data['Date_ordinal'] = [datetime.toordinal(d) for d in data.index]
X_train, X_test, y_train, y_test = train_test_split(data['Date_ordinal'].
```

In [49]:
```
# Create a linear regression model
model = LinearRegression()
```

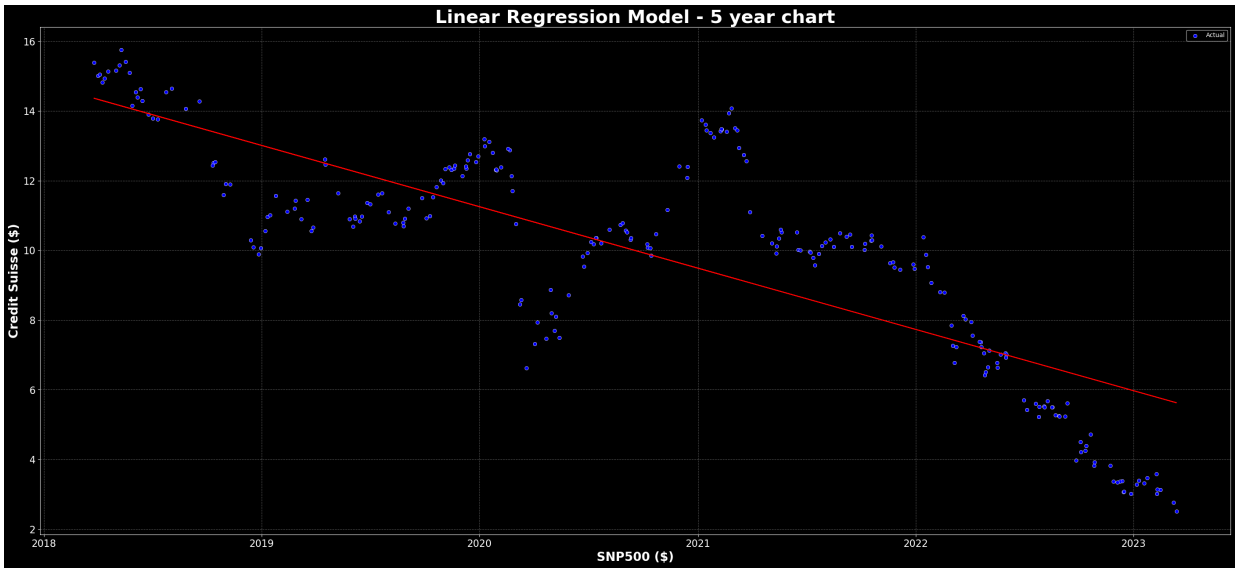
In [50]:
```
# Train the model
model.fit(X_train, y_train)
```

Out[50]:
```
▾ LinearRegression

LinearRegression()
```

In [51]:
```
# Predict the values
y_pred = model.predict(X_test)
```

In [52]:
```
# Convert ordinal value back to date
X_test = [datetime.fromordinal(int(d)) for d in X_test.flatten()]
```

In [53]:
```
plt.style.use('dark_background')
plt.figure(figsize=(35,15))
sns.scatterplot(x=X_test, y=y_test, label='Actual', color='blue')
plt.plot(X_test, y_pred, label='Predicted', color='red')
plt.title('Linear Regression Model - 5 year chart', fontsize=30, color="W
plt.xlabel("SNP500 ($)", fontsize=20, fontweight='bold')
plt.ylabel("Credit Suisse ($)", fontsize=20, fontweight='bold')
plt.tick_params(axis="both", labelsize=16)
plt.grid(linestyle='--', color='gray', alpha=0.7)
plt.show()
```



ONE YEAR TIME SERIES ANALYSIS

In [54]:
```
cs = yf.Ticker("CS")
data = cs.history(period="1y")
```

In [55]:
```
data['Date_ordinal'] = [datetime.toordinal(d) for d in data.index]
X_train, X_test, y_train, y_test = train_test_split(data['Date_ordinal'].
```

In [56]:
```
# Create a linear regression model
model = LinearRegression()
```

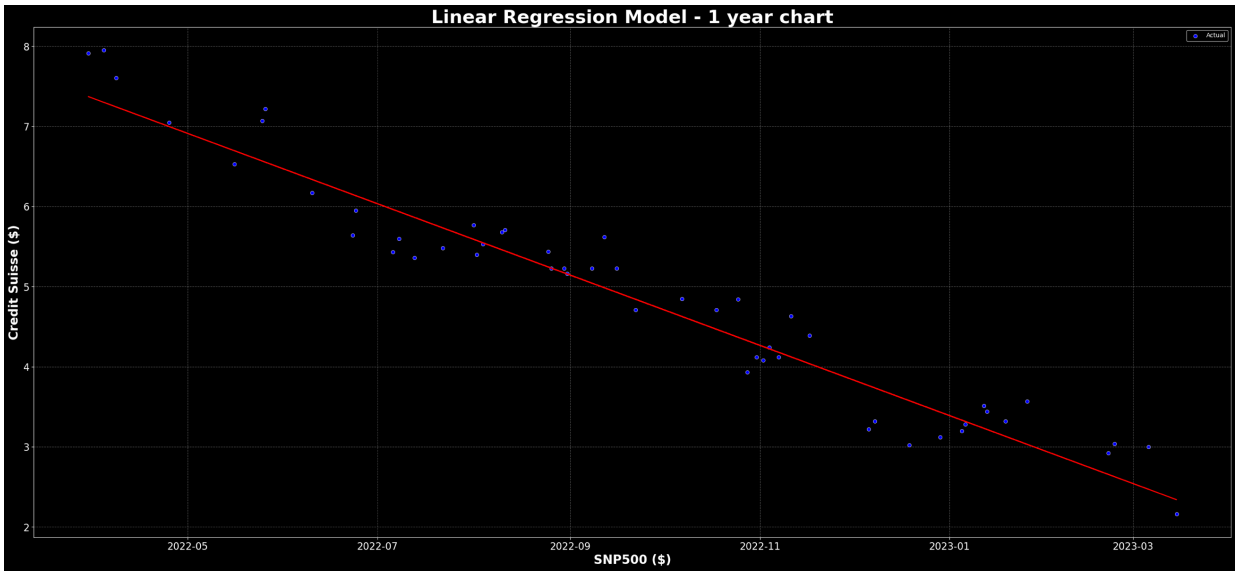
In [57]:
```
# Train the model
model.fit(X_train, y_train)
```

Out[57]:
```
▾ LinearRegression

LinearRegression()
```

In [58]:
```
# Predict the values
y_pred = model.predict(X_test)
```

In [59]:
```
# Convert ordinal value back to date
X_test = [datetime.fromordinal(int(d)) for d in X_test.flatten()]
```

In [60]:
```
plt.style.use('dark_background')
plt.figure(figsize=(35,15))
sns.scatterplot(x=X_test, y=y_test, label='Actual', color='blue')
plt.plot(X_test, y_pred, label='Predicted', color='red')
plt.title('Linear Regression Model - 1 year chart', fontsize=30, color="W
plt.xlabel("SNP500 ($)", fontsize=20, fontweight='bold')
plt.ylabel("Credit Suisse ($)", fontsize=20, fontweight='bold')
plt.tick_params(axis="both", labelsize=16)
plt.grid(linestyle='--', color='gray', alpha=0.7)
plt.show()
```



VOLUME ANALYSIS

In [100…
```
cs = yf.Ticker("CS")
data = cs.history(period="max")
```

In [101…
```
data['Date_ordinal'] = [datetime.toordinal(d) for d in data.index]
X_train, X_test, y_train, y_test = train_test_split(data['Date_ordinal'].
```

In [102…
```
# Create a linear regression model
model = LinearRegression()
```

In [103…
```
# Train the model
model.fit(X_train, y_train)
```

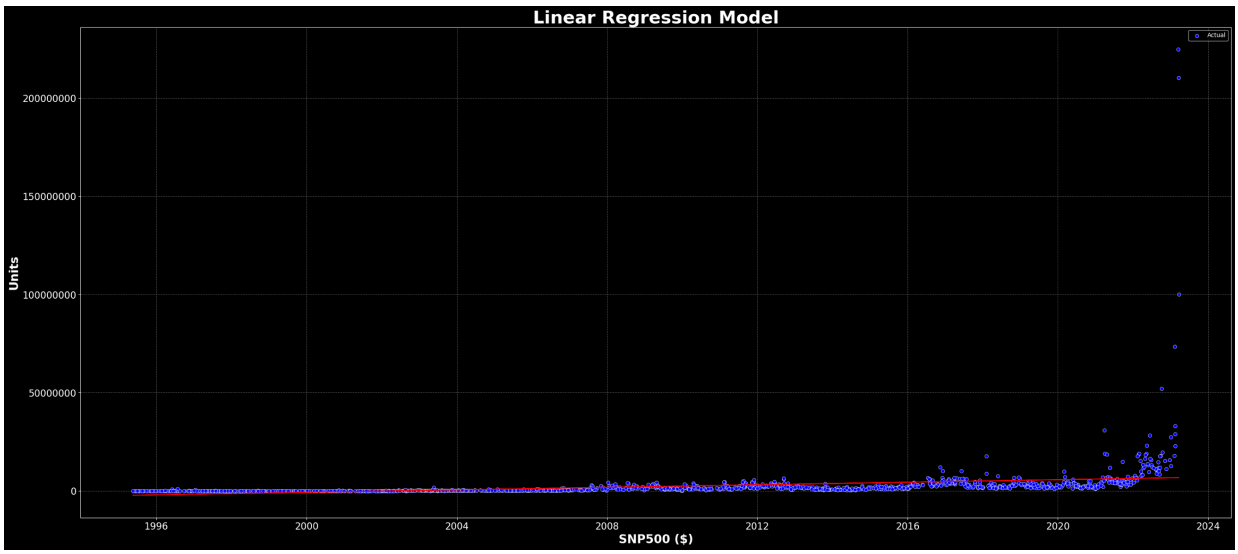
Out[103]:
```
▾ LinearRegression
LinearRegression()
```

In [104…
```
# Predict the values
y_pred = model.predict(X_test)
```

In [105…
```
# Convert ordinal value back to date
X_test = [datetime.fromordinal(int(d)) for d in X_test.flatten()]
```

In [106…
```
from matplotlib.ticker import FuncFormatter
```

In [109…
```
plt.style.use('dark_background')
plt.figure(figsize=(35,15))
plt.gca().yaxis.set_major_formatter(FuncFormatter(lambda x, _: '{:.0f}'.f
sns.scatterplot(x=X_test, y=y_test, label='Actual', color='blue')
plt.plot(X_test, y_pred, label='Predicted', color='red')
plt.title('Linear Regression Model', fontsize=30, color="White", fontweig
plt.xlabel("SNP500 ($)", fontsize=20, fontweight='bold')
plt.ylabel("Units", fontsize=20, fontweight='bold')
plt.tick_params(axis="both", labelsize=16)
plt.grid(linestyle='--', color='gray', alpha=0.7)
plt.show()
```



SNP500 & CREDIT SUISSE

A common approach is to use between 1 to 5 years of historical data to train a linear regression model for stock market prediction. It is important to note that linear regression may not be the most appropriate model for predicting the stock market, as it assumes a linear relationship between the predictors and the target variable, which may not always hold in the complex and dynamic stock market. Linear regression using Machine Learning Library stills show price is moving into downwards directions.

THE UNITS ARE IN DOLLARS FOR SNP500 & CREDIT SUISSE

In [89]:
```
credit_suisse = yf.Ticker("CS").history(period="5y")
snp500 = yf.Ticker("^GSPC").history(period="5y")
```

In [90]:
```
# Preprocess the data by extracting the "Close" prices for Credit Suisse
y = credit_suisse["Close"].values
x = snp500["Close"].values.reshape(-1, 1)
```

In [91]:
```
#Linear Regression Model
model_reg = LinearRegression()
model_reg
```

Out[91]:
```
▾ LinearRegression
LinearRegression()
```

In [92]:
```
#Training Model
model_reg.fit(x,y)
```

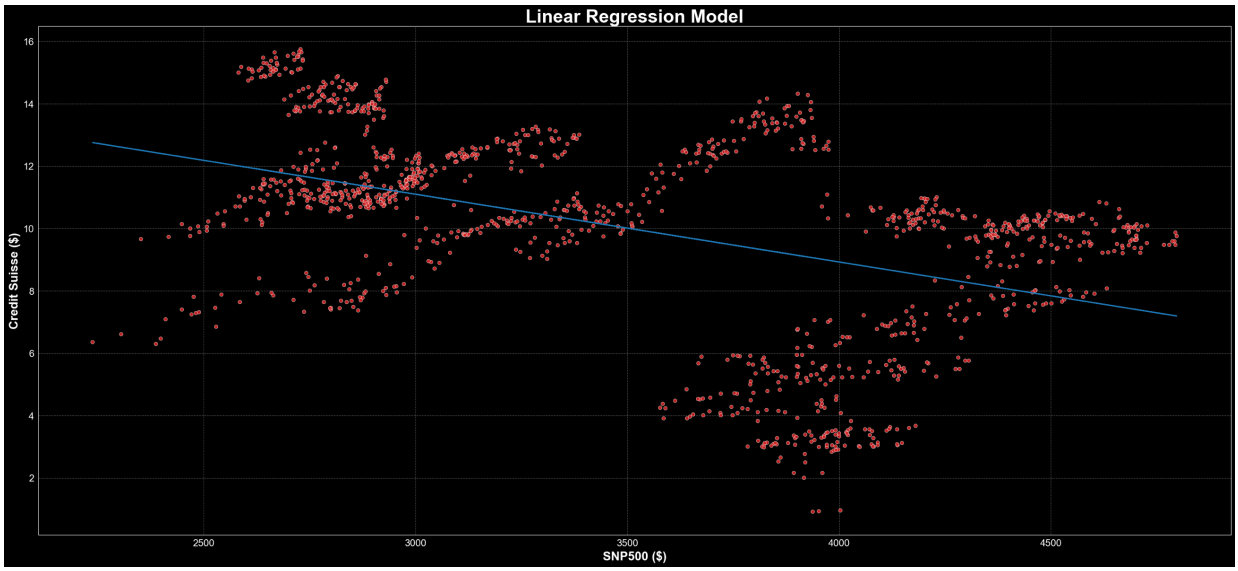
Out[92]:
```
▾ LinearRegression
LinearRegression()
```

In [93]:
```
pred_y = model_reg.predict(x)
```

In [94]:
```
pred_y.shape
```

Out[94]:
```
(1258,)
```

In [95]:
```
plt.style.use('dark_background')
plt.figure(figsize=(35,15))
sns.scatterplot(x=x.flatten(), y=y.flatten(), color='tab:red', alpha=0.9)
plt.plot(x, pred_y, color='tab:blue', linewidth=2)
plt.title('Linear Regression Model', fontsize=30, color="White", fontweig
plt.xlabel("SNP500 ($)", fontsize=20, fontweight='bold')
plt.ylabel("Credit Suisse ($)", fontsize=20, fontweight='bold')
plt.tick_params(axis="both", labelsize=16)
plt.grid(linestyle='--', color='gray', alpha=0.7)
plt.show()
```



In [96]:
```
from sklearn.metrics import mean_squared_error
```

In [97]:
```
mse = mean_squared_error(y, pred_y)
print("Mean squared error:", mse)
```

```
Mean squared error: 7.820189149817397
```

In [98]:
```
# calculate the mean squared error
mse = np.mean((y - pred_y)**2)
print("Mean Squared Error:", mse)
```

```
Mean Squared Error: 7.820189149817397
```

The mean squared error (MSE) value of 7.777455434440157 is a measure of the average squared differences between the actual values of the Credit Suisse stock prices and the predicted values of the stock prices based on the S&P 500 index.

This indicates that the model used to predict the Credit Suisse stock prices based on the S&P 500 index may have some degree of error or inaccuracy in its predictions. The MSE value of 7.777455434440157 can be used as a reference for evaluating and comparing the performance of other predictive models for the same task.

It's difficult to say whether a mean squared error of 7.777455434440157 is good or bad without more context. Generally, a lower mean squared error indicates that the model is better at making predictions.

However, what is considered a "good" mean squared error can vary depending on the specific problem and the data being used. It's important to compare the mean squared error of a model to other models and to consider the practical implications of the error.

FTSE100 & CREDIT SUISSE

THE UNITS ARE IN POUNDS FOR FTSE100 & CREDIT SUISSE

In [99]:
```
credit_suisse = yf.Ticker("CS").history(period="5y")
ftse100 = yf.Ticker("^FTSE").history(period="5y")
```

In [100…
```
from forex_python.converter import CurrencyRates
```

In [101…
```
c = CurrencyRates()
exchange_rate = c.get_rate('USD', 'GBP')
credit_suisse["Close"] = credit_suisse["Close"] * exchange_rate
credit_suisse["Open"] = credit_suisse["Open"] * exchange_rate
credit_suisse["High"] = credit_suisse["High"] * exchange_rate
credit_suisse["Low"] = credit_suisse["Low"] * exchange_rate
```

In [102…
```
credit_suisse.tail(1)
```

Out[102]:

Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
2023-03-22 00:00:00-04:00	0.76792	0.784258	0.743412	0.751581	100043800	0.0	0.0

In [103…
```
ftse100.tail(1)
```

Out[103]:

Date	Open	High	Low	Close	Volume	Divider
2023-03-23 00:00:00+00:00	7566.839844	7566.839844	7476.129883	7487.700195	0	

In [104…
```
# Preprocess the data by extracting the "Close" prices for Credit Suisse
y = credit_suisse["Close"].values
x = ftse100["Close"].values.reshape(-1, 1)
```

In [105…
```
y #units are on pounds (credit suisse)
```

Out[105]:
```
array([12.40990942, 12.57388463, 12.33537459, ...,  0.76791963,
        0.79242773,  0.75158093])
```

In [106…
```
y.shape
```

Out[106]: (1258,)

In [107…
```
x.shape
```

Out[107]: (1262, 1)

In [108…
```
x = x[4:]
```

In [109…
```
x.shape
```

Out[109]: (1258, 1)

In [110…
```
#Linear Regression Model
model_reg = LinearRegression()
model_reg
```

Out[110]:
```
▾ LinearRegression
LinearRegression()
```

In [111…
```
#Training Model
model_reg.fit(x,y)
```

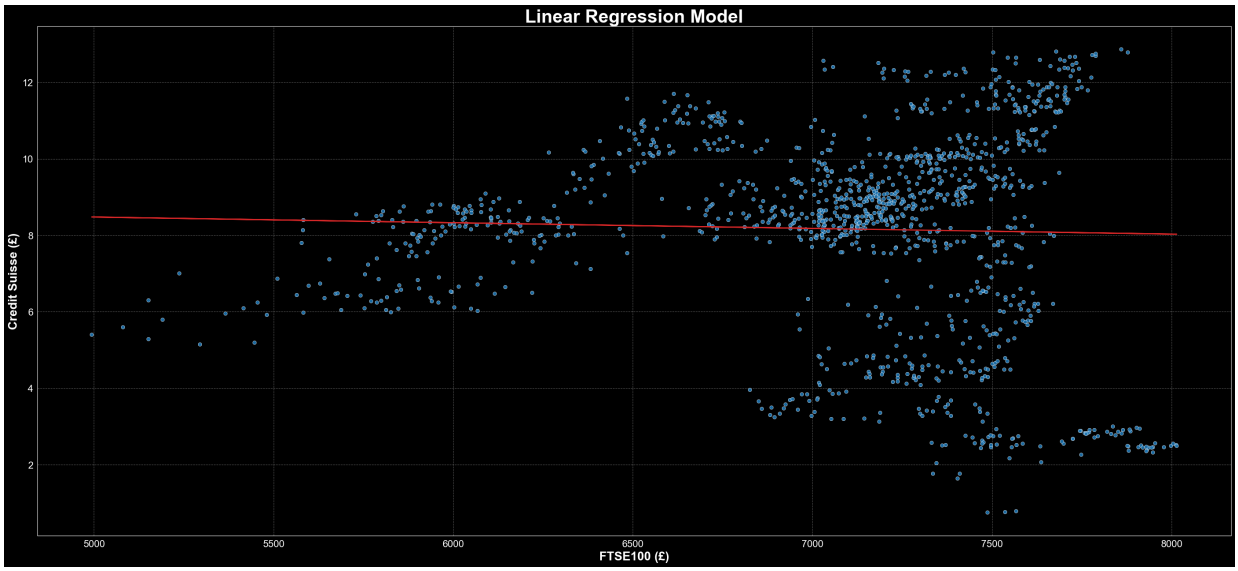
Out[111]:
```
▾ LinearRegression
LinearRegression()
```

In [112…
```
pred_y = model_reg.predict(x)
```

In [113…
```
pred_y.shape
```

Out[113]: (1258,)

In [114…
```
plt.style.use('dark_background')
plt.figure(figsize=(35,15))
sns.scatterplot(x=x.flatten(), y=y.flatten(), color='tab:blue', alpha=0.9
plt.plot(x, pred_y, color='tab:red', linewidth=2)
plt.title('Linear Regression Model', fontsize=30, color="White", fontweig
plt.xlabel("FTSE100 (£)", fontsize=20, fontweight='bold')
plt.ylabel("Credit Suisse (£)", fontsize=20, fontweight='bold')
plt.tick_params(axis="both", labelsize=16)
plt.grid(linestyle='--', color='gray', alpha=0.7)
plt.show()
```



In [115…
```
mse = mean_squared_error(y, pred_y)
print("Mean squared error:", mse)
```

```
Mean squared error: 6.5845920220289385
```

In [116…
```
# calculate the mean squared error
mse = np.mean((y - pred_y)**2)
print("Mean Squared Error:", mse)
```

```
Mean Squared Error: 6.5845920220289385
```

NEURAL NETWORK REGRESSION

CREDIT SUISSE & SNP500

In [143…
```
credit_suisse = yf.Ticker("CS").history(period="5y")
snp500 = yf.Ticker("^GSPC").history(period="5y")
```

In [144…
```
# Preprocess the data by extracting the "Close" prices for Credit Suisse
y = credit_suisse["Close"].values
x = snp500["Close"].values.reshape(-1, 1)
```

In [145…
```
# Preprocess the data
print(y.shape)
print(x.shape)
```

```
(1258,)
(1258, 1)
```

In [146…
```
from keras.layers import Dense, Dropout
```

In [147…
```
# Define activation and dropout parameters
activation = 'relu'
dropout_rate = 0.2

# Initialize the neural network model
model = Sequential()

# Add layers with specified architecture
for layer_size in [16, 32, 64, 128, 256]:
    model.add(Dense(layer_size, input_shape=(1,), activation=activation))
    model.add(Dropout(dropout_rate))

# Add output layer with linear activation
model.add(Dense(1, activation='linear'))
```

In [148…
```
# Compile the model with the specified loss function and optimizer
model.compile(loss='mean_squared_error', optimizer='adam', metrics=['mean
# Train the model
model.fit(x, y, epochs=250, batch_size=32, verbose=1)
```

```
Epoch 1/250
40/40 [==============================] - 1s 3ms/step - loss: 13075.5029
- mean_squared_error: 13075.5029
Epoch 2/250
40/40 [==============================] - 0s 3ms/step - loss: 2591.7158 -
mean_squared_error: 2591.7158
Epoch 3/250
40/40 [==============================] - 0s 3ms/step - loss: 1449.1232 -
mean_squared_error: 1449.1232
Epoch 4/250
40/40 [==============================] - 0s 3ms/step - loss: 874.3273 -
mean_squared_error: 874.3273
Epoch 5/250
40/40 [==============================] - 0s 3ms/step - loss: 707.9475 -
mean_squared_error: 707.9475
Epoch 6/250
40/40 [==============================] - 0s 2ms/step - loss: 539.0250 -
mean_squared_error: 539.0250
Epoch 7/250
40/40 [==============================] - 0s 4ms/step - loss: 421.0649 -
```

```
mean_squared_error: 421.0649
Epoch 8/250
40/40 [==============================] - 0s 2ms/step - loss: 343.3991 -
mean_squared_error: 343.3991
Epoch 9/250
40/40 [==============================] - 0s 2ms/step - loss: 293.5563 -
mean_squared_error: 293.5563
Epoch 10/250
40/40 [==============================] - 0s 2ms/step - loss: 265.3694 -
mean_squared_error: 265.3694
Epoch 11/250
40/40 [==============================] - 0s 2ms/step - loss: 227.7009 -
mean_squared_error: 227.7009
Epoch 12/250
40/40 [==============================] - 0s 2ms/step - loss: 194.1156 -
mean_squared_error: 194.1156
Epoch 13/250
40/40 [==============================] - 0s 3ms/step - loss: 194.5367 -
mean_squared_error: 194.5367
Epoch 14/250
40/40 [==============================] - 0s 5ms/step - loss: 147.1288 -
mean_squared_error: 147.1288
Epoch 15/250
40/40 [==============================] - 0s 2ms/step - loss: 152.5828 -
mean_squared_error: 152.5828
Epoch 16/250
40/40 [==============================] - 0s 2ms/step - loss: 129.1334 -
mean_squared_error: 129.1334
Epoch 17/250
40/40 [==============================] - 0s 2ms/step - loss: 130.6100 -
mean_squared_error: 130.6100
Epoch 18/250
40/40 [==============================] - 0s 3ms/step - loss: 110.5337 -
mean_squared_error: 110.5337
Epoch 19/250
40/40 [==============================] - 0s 2ms/step - loss: 102.0994 -
mean_squared_error: 102.0994
Epoch 20/250
40/40 [==============================] - 0s 2ms/step - loss: 109.7932 -
mean_squared_error: 109.7932
Epoch 21/250
40/40 [==============================] - 0s 2ms/step - loss: 87.4296 - m
ean_squared_error: 87.4296
Epoch 22/250
40/40 [==============================] - 0s 2ms/step - loss: 84.9521 - m
ean_squared_error: 84.9521
Epoch 23/250
40/40 [==============================] - 0s 2ms/step - loss: 80.3151 - m
ean_squared_error: 80.3151
Epoch 24/250
40/40 [==============================] - 0s 2ms/step - loss: 79.6451 - m
ean_squared_error: 79.6451
Epoch 25/250
40/40 [==============================] - 0s 3ms/step - loss: 79.0324 - m
ean_squared_error: 79.0324
Epoch 26/250
40/40 [==============================] - 0s 3ms/step - loss: 70.1860 - m
```

```
ean_squared_error: 70.1860
Epoch 27/250
40/40 [==============================] - 0s 3ms/step - loss: 68.1494 - m
ean_squared_error: 68.1494
Epoch 28/250
40/40 [==============================] - 0s 3ms/step - loss: 63.8184 - m
ean_squared_error: 63.8184
Epoch 29/250
40/40 [==============================] - 0s 2ms/step - loss: 65.1780 - m
ean_squared_error: 65.1780
Epoch 30/250
40/40 [==============================] - 0s 3ms/step - loss: 55.1391 - m
ean_squared_error: 55.1391
Epoch 31/250
40/40 [==============================] - 0s 3ms/step - loss: 58.4757 - m
ean_squared_error: 58.4757
Epoch 32/250
40/40 [==============================] - 0s 2ms/step - loss: 54.3021 - m
ean_squared_error: 54.3021
Epoch 33/250
40/40 [==============================] - 0s 2ms/step - loss: 52.0620 - m
ean_squared_error: 52.0620
Epoch 34/250
40/40 [==============================] - 0s 2ms/step - loss: 50.5989 - m
ean_squared_error: 50.5989
Epoch 35/250
40/40 [==============================] - 0s 2ms/step - loss: 47.4356 - m
ean_squared_error: 47.4356
Epoch 36/250
40/40 [==============================] - 0s 2ms/step - loss: 48.7417 - m
ean_squared_error: 48.7417
Epoch 37/250
40/40 [==============================] - 0s 2ms/step - loss: 45.0190 - m
ean_squared_error: 45.0190
Epoch 38/250
40/40 [==============================] - 0s 2ms/step - loss: 41.9719 - m
ean_squared_error: 41.9719
Epoch 39/250
40/40 [==============================] - 0s 3ms/step - loss: 43.3884 - m
ean_squared_error: 43.3884
Epoch 40/250
40/40 [==============================] - 0s 2ms/step - loss: 44.4293 - m
ean_squared_error: 44.4293
Epoch 41/250
40/40 [==============================] - 0s 3ms/step - loss: 39.4684 - m
ean_squared_error: 39.4684
Epoch 42/250
40/40 [==============================] - 0s 2ms/step - loss: 40.5420 - m
ean_squared_error: 40.5420
Epoch 43/250
40/40 [==============================] - 0s 3ms/step - loss: 38.6888 - m
ean_squared_error: 38.6888
Epoch 44/250
40/40 [==============================] - 0s 2ms/step - loss: 35.9611 - m
ean_squared_error: 35.9611
Epoch 45/250
40/40 [==============================] - 0s 3ms/step - loss: 37.0057 - m
```

```
ean_squared_error: 37.0057
Epoch 46/250
40/40 [==============================] - 0s 2ms/step - loss: 37.4800 - m
ean_squared_error: 37.4800
Epoch 47/250
40/40 [==============================] - 0s 2ms/step - loss: 34.9991 - m
ean_squared_error: 34.9991
Epoch 48/250
40/40 [==============================] - 0s 2ms/step - loss: 35.2491 - m
ean_squared_error: 35.2491
Epoch 49/250
40/40 [==============================] - 0s 3ms/step - loss: 33.4743 - m
ean_squared_error: 33.4743
Epoch 50/250
40/40 [==============================] - 0s 2ms/step - loss: 34.2797 - m
ean_squared_error: 34.2797
Epoch 51/250
40/40 [==============================] - 0s 3ms/step - loss: 31.5830 - m
ean_squared_error: 31.5830
Epoch 52/250
40/40 [==============================] - 0s 3ms/step - loss: 31.3733 - m
ean_squared_error: 31.3733
Epoch 53/250
40/40 [==============================] - 0s 2ms/step - loss: 30.3685 - m
ean_squared_error: 30.3685
Epoch 54/250
40/40 [==============================] - 0s 2ms/step - loss: 30.6050 - m
ean_squared_error: 30.6050
Epoch 55/250
40/40 [==============================] - 0s 2ms/step - loss: 31.4028 - m
ean_squared_error: 31.4028
Epoch 56/250
40/40 [==============================] - 0s 2ms/step - loss: 31.4509 - m
ean_squared_error: 31.4509
Epoch 57/250
40/40 [==============================] - 0s 3ms/step - loss: 29.1246 - m
ean_squared_error: 29.1246
Epoch 58/250
40/40 [==============================] - 0s 2ms/step - loss: 26.3616 - m
ean_squared_error: 26.3616
Epoch 59/250
40/40 [==============================] - 0s 2ms/step - loss: 27.5851 - m
ean_squared_error: 27.5851
Epoch 60/250
40/40 [==============================] - 0s 2ms/step - loss: 27.0646 - m
ean_squared_error: 27.0646
Epoch 61/250
40/40 [==============================] - 0s 2ms/step - loss: 26.3113 - m
ean_squared_error: 26.3113
Epoch 62/250
40/40 [==============================] - 0s 2ms/step - loss: 29.0959 - m
ean_squared_error: 29.0959
Epoch 63/250
40/40 [==============================] - 0s 3ms/step - loss: 25.5723 - m
ean_squared_error: 25.5723
Epoch 64/250
40/40 [==============================] - 0s 2ms/step - loss: 26.4503 - m
```

```
ean_squared_error: 26.4503
Epoch 65/250
40/40 [==============================] - 0s 2ms/step - loss: 26.5493 - m
ean_squared_error: 26.5493
Epoch 66/250
40/40 [==============================] - 0s 3ms/step - loss: 24.8261 - m
ean_squared_error: 24.8261
Epoch 67/250
40/40 [==============================] - 0s 2ms/step - loss: 26.1670 - m
ean_squared_error: 26.1670
Epoch 68/250
40/40 [==============================] - 0s 2ms/step - loss: 25.4118 - m
ean_squared_error: 25.4118
Epoch 69/250
40/40 [==============================] - 0s 2ms/step - loss: 25.8555 - m
ean_squared_error: 25.8555
Epoch 70/250
40/40 [==============================] - 0s 3ms/step - loss: 26.4481 - m
ean_squared_error: 26.4481
Epoch 71/250
40/40 [==============================] - 0s 2ms/step - loss: 24.0890 - m
ean_squared_error: 24.0890
Epoch 72/250
40/40 [==============================] - 0s 2ms/step - loss: 24.9015 - m
ean_squared_error: 24.9015
Epoch 73/250
40/40 [==============================] - 0s 2ms/step - loss: 25.6150 - m
ean_squared_error: 25.6150
Epoch 74/250
40/40 [==============================] - 0s 2ms/step - loss: 24.6986 - m
ean_squared_error: 24.6986
Epoch 75/250
40/40 [==============================] - 0s 3ms/step - loss: 25.1208 - m
ean_squared_error: 25.1208
Epoch 76/250
40/40 [==============================] - 0s 2ms/step - loss: 24.5107 - m
ean_squared_error: 24.5107
Epoch 77/250
40/40 [==============================] - 0s 2ms/step - loss: 23.9236 - m
ean_squared_error: 23.9236
Epoch 78/250
40/40 [==============================] - 0s 2ms/step - loss: 23.5507 - m
ean_squared_error: 23.5507
Epoch 79/250
40/40 [==============================] - 0s 3ms/step - loss: 24.2132 - m
ean_squared_error: 24.2132
Epoch 80/250
40/40 [==============================] - 0s 2ms/step - loss: 23.8153 - m
ean_squared_error: 23.8153
Epoch 81/250
40/40 [==============================] - 0s 3ms/step - loss: 22.4952 - m
ean_squared_error: 22.4952
Epoch 82/250
40/40 [==============================] - 0s 2ms/step - loss: 23.1272 - m
ean_squared_error: 23.1272
Epoch 83/250
40/40 [==============================] - 0s 3ms/step - loss: 23.0064 - m
```

```
ean_squared_error: 23.0064
Epoch 84/250
40/40 [==============================] - 0s 3ms/step - loss: 23.9996 - m
ean_squared_error: 23.9996
Epoch 85/250
40/40 [==============================] - 0s 2ms/step - loss: 24.0745 - m
ean_squared_error: 24.0745
Epoch 86/250
40/40 [==============================] - 0s 3ms/step - loss: 22.6452 - m
ean_squared_error: 22.6452
Epoch 87/250
40/40 [==============================] - 0s 2ms/step - loss: 21.3872 - m
ean_squared_error: 21.3872
Epoch 88/250
40/40 [==============================] - 0s 5ms/step - loss: 22.6239 - m
ean_squared_error: 22.6239
Epoch 89/250
40/40 [==============================] - 0s 2ms/step - loss: 21.8676 - m
ean_squared_error: 21.8676
Epoch 90/250
40/40 [==============================] - 0s 2ms/step - loss: 22.3085 - m
ean_squared_error: 22.3085
Epoch 91/250
40/40 [==============================] - 0s 3ms/step - loss: 21.3830 - m
ean_squared_error: 21.3830
Epoch 92/250
40/40 [==============================] - 0s 2ms/step - loss: 22.2698 - m
ean_squared_error: 22.2698
Epoch 93/250
40/40 [==============================] - 0s 2ms/step - loss: 23.1638 - m
ean_squared_error: 23.1638
Epoch 94/250
40/40 [==============================] - 0s 2ms/step - loss: 21.5814 - m
ean_squared_error: 21.5814
Epoch 95/250
40/40 [==============================] - 0s 2ms/step - loss: 21.2446 - m
ean_squared_error: 21.2446
Epoch 96/250
40/40 [==============================] - 0s 3ms/step - loss: 20.4182 - m
ean_squared_error: 20.4182
Epoch 97/250
40/40 [==============================] - 0s 2ms/step - loss: 21.4031 - m
ean_squared_error: 21.4031
Epoch 98/250
40/40 [==============================] - 0s 2ms/step - loss: 21.0020 - m
ean_squared_error: 21.0020
Epoch 99/250
40/40 [==============================] - 0s 2ms/step - loss: 20.5760 - m
ean_squared_error: 20.5760
Epoch 100/250
40/40 [==============================] - 0s 2ms/step - loss: 21.0659 - m
ean_squared_error: 21.0659
Epoch 101/250
40/40 [==============================] - 0s 2ms/step - loss: 20.3631 - m
ean_squared_error: 20.3631
Epoch 102/250
40/40 [==============================] - 0s 2ms/step - loss: 20.5500 - m
```

```
ean_squared_error: 20.5500
Epoch 103/250
40/40 [==============================] - 0s 3ms/step - loss: 20.7849 - m
ean_squared_error: 20.7849
Epoch 104/250
40/40 [==============================] - 0s 2ms/step - loss: 19.7230 - m
ean_squared_error: 19.7230
Epoch 105/250
40/40 [==============================] - 0s 3ms/step - loss: 20.3916 - m
ean_squared_error: 20.3916
Epoch 106/250
40/40 [==============================] - 0s 2ms/step - loss: 20.8305 - m
ean_squared_error: 20.8305
Epoch 107/250
40/40 [==============================] - 0s 2ms/step - loss: 20.5666 - m
ean_squared_error: 20.5666
Epoch 108/250
40/40 [==============================] - 0s 2ms/step - loss: 21.1236 - m
ean_squared_error: 21.1236
Epoch 109/250
40/40 [==============================] - 0s 2ms/step - loss: 20.3115 - m
ean_squared_error: 20.3115
Epoch 110/250
40/40 [==============================] - 0s 2ms/step - loss: 19.9997 - m
ean_squared_error: 19.9997
Epoch 111/250
40/40 [==============================] - 0s 2ms/step - loss: 20.5746 - m
ean_squared_error: 20.5746
Epoch 112/250
40/40 [==============================] - 0s 3ms/step - loss: 19.5773 - m
ean_squared_error: 19.5773
Epoch 113/250
40/40 [==============================] - 0s 2ms/step - loss: 19.1643 - m
ean_squared_error: 19.1643
Epoch 114/250
40/40 [==============================] - 0s 2ms/step - loss: 19.2199 - m
ean_squared_error: 19.2199
Epoch 115/250
40/40 [==============================] - 0s 2ms/step - loss: 18.7475 - m
ean_squared_error: 18.7475
Epoch 116/250
40/40 [==============================] - 0s 2ms/step - loss: 19.3550 - m
ean_squared_error: 19.3550
Epoch 117/250
40/40 [==============================] - 0s 3ms/step - loss: 18.8713 - m
ean_squared_error: 18.8713
Epoch 118/250
40/40 [==============================] - 0s 3ms/step - loss: 18.9559 - m
ean_squared_error: 18.9559
Epoch 119/250
40/40 [==============================] - 0s 3ms/step - loss: 19.7615 - m
ean_squared_error: 19.7615
Epoch 120/250
40/40 [==============================] - 0s 2ms/step - loss: 19.4535 - m
ean_squared_error: 19.4535
Epoch 121/250
40/40 [==============================] - 0s 2ms/step - loss: 19.6416 - m
```

```
ean_squared_error: 19.6416
Epoch 122/250
40/40 [==============================] - 0s 3ms/step - loss: 20.0548 - m
ean_squared_error: 20.0548
Epoch 123/250
40/40 [==============================] - 0s 2ms/step - loss: 19.0815 - m
ean_squared_error: 19.0815
Epoch 124/250
40/40 [==============================] - 0s 3ms/step - loss: 18.5069 - m
ean_squared_error: 18.5069
Epoch 125/250
40/40 [==============================] - 0s 2ms/step - loss: 18.0554 - m
ean_squared_error: 18.0554
Epoch 126/250
40/40 [==============================] - 0s 2ms/step - loss: 17.7406 - m
ean_squared_error: 17.7406
Epoch 127/250
40/40 [==============================] - 0s 2ms/step - loss: 18.4253 - m
ean_squared_error: 18.4253
Epoch 128/250
40/40 [==============================] - 0s 2ms/step - loss: 18.7018 - m
ean_squared_error: 18.7018
Epoch 129/250
40/40 [==============================] - 0s 3ms/step - loss: 18.1460 - m
ean_squared_error: 18.1460
Epoch 130/250
40/40 [==============================] - 0s 2ms/step - loss: 18.3571 - m
ean_squared_error: 18.3571
Epoch 131/250
40/40 [==============================] - 0s 3ms/step - loss: 18.0887 - m
ean_squared_error: 18.0887
Epoch 132/250
40/40 [==============================] - 0s 2ms/step - loss: 18.6994 - m
ean_squared_error: 18.6994
Epoch 133/250
40/40 [==============================] - 0s 3ms/step - loss: 18.1112 - m
ean_squared_error: 18.1112
Epoch 134/250
40/40 [==============================] - 0s 2ms/step - loss: 17.2139 - m
ean_squared_error: 17.2139
Epoch 135/250
40/40 [==============================] - 0s 4ms/step - loss: 18.0543 - m
ean_squared_error: 18.0543
Epoch 136/250
40/40 [==============================] - 0s 2ms/step - loss: 18.6478 - m
ean_squared_error: 18.6478
Epoch 137/250
40/40 [==============================] - 0s 2ms/step - loss: 17.7722 - m
ean_squared_error: 17.7722
Epoch 138/250
40/40 [==============================] - 0s 3ms/step - loss: 17.3391 - m
ean_squared_error: 17.3391
Epoch 139/250
40/40 [==============================] - 0s 2ms/step - loss: 18.6678 - m
ean_squared_error: 18.6678
Epoch 140/250
40/40 [==============================] - 0s 2ms/step - loss: 18.1770 - m
```

```
ean_squared_error: 18.1770
Epoch 141/250
40/40 [==============================] - 0s 3ms/step - loss: 17.1418 - m
ean_squared_error: 17.1418
Epoch 142/250
40/40 [==============================] - 0s 3ms/step - loss: 17.3855 - m
ean_squared_error: 17.3855
Epoch 143/250
40/40 [==============================] - 0s 3ms/step - loss: 17.7665 - m
ean_squared_error: 17.7665
Epoch 144/250
40/40 [==============================] - 0s 2ms/step - loss: 17.1913 - m
ean_squared_error: 17.1913
Epoch 145/250
40/40 [==============================] - 0s 2ms/step - loss: 17.3754 - m
ean_squared_error: 17.3754
Epoch 146/250
40/40 [==============================] - 0s 3ms/step - loss: 17.7638 - m
ean_squared_error: 17.7638
Epoch 147/250
40/40 [==============================] - 0s 2ms/step - loss: 16.1936 - m
ean_squared_error: 16.1936
Epoch 148/250
40/40 [==============================] - 0s 2ms/step - loss: 17.2552 - m
ean_squared_error: 17.2552
Epoch 149/250
40/40 [==============================] - 0s 3ms/step - loss: 16.9114 - m
ean_squared_error: 16.9114
Epoch 150/250
40/40 [==============================] - 0s 2ms/step - loss: 16.3241 - m
ean_squared_error: 16.3241
Epoch 151/250
40/40 [==============================] - 0s 2ms/step - loss: 16.8424 - m
ean_squared_error: 16.8424
Epoch 152/250
40/40 [==============================] - 0s 3ms/step - loss: 16.1001 - m
ean_squared_error: 16.1001
Epoch 153/250
40/40 [==============================] - 0s 2ms/step - loss: 16.6930 - m
ean_squared_error: 16.6930
Epoch 154/250
40/40 [==============================] - 0s 2ms/step - loss: 16.3176 - m
ean_squared_error: 16.3176
Epoch 155/250
40/40 [==============================] - 0s 2ms/step - loss: 16.3215 - m
ean_squared_error: 16.3215
Epoch 156/250
40/40 [==============================] - 0s 2ms/step - loss: 16.5621 - m
ean_squared_error: 16.5621
Epoch 157/250
40/40 [==============================] - 0s 2ms/step - loss: 15.3555 - m
ean_squared_error: 15.3555
Epoch 158/250
40/40 [==============================] - 0s 2ms/step - loss: 15.7109 - m
ean_squared_error: 15.7109
Epoch 159/250
40/40 [==============================] - 0s 2ms/step - loss: 15.3414 - m
```

```
ean_squared_error: 15.3414
Epoch 160/250
40/40 [==============================] - 0s 3ms/step - loss: 16.2644 - m
ean_squared_error: 16.2644
Epoch 161/250
40/40 [==============================] - 0s 2ms/step - loss: 16.1623 - m
ean_squared_error: 16.1623
Epoch 162/250
40/40 [==============================] - 0s 3ms/step - loss: 15.4206 - m
ean_squared_error: 15.4206
Epoch 163/250
40/40 [==============================] - 0s 2ms/step - loss: 14.7894 - m
ean_squared_error: 14.7894
Epoch 164/250
40/40 [==============================] - 0s 2ms/step - loss: 15.1239 - m
ean_squared_error: 15.1239
Epoch 165/250
40/40 [==============================] - 0s 3ms/step - loss: 14.7255 - m
ean_squared_error: 14.7255
Epoch 166/250
40/40 [==============================] - 0s 2ms/step - loss: 14.4155 - m
ean_squared_error: 14.4155
Epoch 167/250
40/40 [==============================] - 0s 3ms/step - loss: 13.7582 - m
ean_squared_error: 13.7582
Epoch 168/250
40/40 [==============================] - 0s 2ms/step - loss: 14.1769 - m
ean_squared_error: 14.1769
Epoch 169/250
40/40 [==============================] - 0s 2ms/step - loss: 14.0338 - m
ean_squared_error: 14.0338
Epoch 170/250
40/40 [==============================] - 0s 3ms/step - loss: 14.1833 - m
ean_squared_error: 14.1833
Epoch 171/250
40/40 [==============================] - 0s 3ms/step - loss: 14.1001 - m
ean_squared_error: 14.1001
Epoch 172/250
40/40 [==============================] - 0s 2ms/step - loss: 13.2150 - m
ean_squared_error: 13.2150
Epoch 173/250
40/40 [==============================] - 0s 2ms/step - loss: 13.2836 - m
ean_squared_error: 13.2836
Epoch 174/250
40/40 [==============================] - 0s 3ms/step - loss: 12.9355 - m
ean_squared_error: 12.9355
Epoch 175/250
40/40 [==============================] - 0s 2ms/step - loss: 12.5637 - m
ean_squared_error: 12.5637
Epoch 176/250
40/40 [==============================] - 0s 2ms/step - loss: 12.3646 - m
ean_squared_error: 12.3646
Epoch 177/250
40/40 [==============================] - 0s 2ms/step - loss: 13.0804 - m
ean_squared_error: 13.0804
Epoch 178/250
40/40 [==============================] - 0s 3ms/step - loss: 12.1845 - m
```

```
ean_squared_error: 12.1845
Epoch 179/250
40/40 [==============================] - 0s 2ms/step - loss: 11.9448 - m
ean_squared_error: 11.9448
Epoch 180/250
40/40 [==============================] - 0s 2ms/step - loss: 12.2040 - m
ean_squared_error: 12.2040
Epoch 181/250
40/40 [==============================] - 0s 2ms/step - loss: 11.9123 - m
ean_squared_error: 11.9123
Epoch 182/250
40/40 [==============================] - 0s 3ms/step - loss: 11.7184 - m
ean_squared_error: 11.7184
Epoch 183/250
40/40 [==============================] - 0s 2ms/step - loss: 11.3563 - m
ean_squared_error: 11.3563
Epoch 184/250
40/40 [==============================] - 0s 2ms/step - loss: 11.3206 - m
ean_squared_error: 11.3206
Epoch 185/250
40/40 [==============================] - 0s 2ms/step - loss: 11.2025 - m
ean_squared_error: 11.2025
Epoch 186/250
40/40 [==============================] - 0s 2ms/step - loss: 11.2248 - m
ean_squared_error: 11.2248
Epoch 187/250
40/40 [==============================] - 0s 2ms/step - loss: 10.9849 - m
ean_squared_error: 10.9849
Epoch 188/250
40/40 [==============================] - 0s 2ms/step - loss: 10.7695 - m
ean_squared_error: 10.7695
Epoch 189/250
40/40 [==============================] - 0s 2ms/step - loss: 10.6082 - m
ean_squared_error: 10.6082
Epoch 190/250
40/40 [==============================] - 0s 2ms/step - loss: 10.9165 - m
ean_squared_error: 10.9165
Epoch 191/250
40/40 [==============================] - 0s 3ms/step - loss: 10.7122 - m
ean_squared_error: 10.7122
Epoch 192/250
40/40 [==============================] - 0s 2ms/step - loss: 10.4910 - m
ean_squared_error: 10.4910
Epoch 193/250
40/40 [==============================] - 0s 5ms/step - loss: 10.6161 - m
ean_squared_error: 10.6161
Epoch 194/250
40/40 [==============================] - 0s 2ms/step - loss: 10.6935 - m
ean_squared_error: 10.6935
Epoch 195/250
40/40 [==============================] - 0s 2ms/step - loss: 10.5589 - m
ean_squared_error: 10.5589
Epoch 196/250
40/40 [==============================] - 0s 3ms/step - loss: 10.3499 - m
ean_squared_error: 10.3499
Epoch 197/250
40/40 [==============================] - 0s 3ms/step - loss: 10.3846 - m
```

```
ean_squared_error: 10.3846
Epoch 198/250
40/40 [==============================] - 0s 2ms/step - loss: 10.5009 - m
ean_squared_error: 10.5009
Epoch 199/250
40/40 [==============================] - 0s 2ms/step - loss: 10.4051 - m
ean_squared_error: 10.4051
Epoch 200/250
40/40 [==============================] - 0s 3ms/step - loss: 10.3710 - m
ean_squared_error: 10.3710
Epoch 201/250
40/40 [==============================] - 0s 3ms/step - loss: 10.3096 - m
ean_squared_error: 10.3096
Epoch 202/250
40/40 [==============================] - 0s 3ms/step - loss: 10.5788 - m
ean_squared_error: 10.5788
Epoch 203/250
40/40 [==============================] - 0s 4ms/step - loss: 10.2241 - m
ean_squared_error: 10.2241
Epoch 204/250
40/40 [==============================] - 0s 3ms/step - loss: 10.0352 - m
ean_squared_error: 10.0352
Epoch 205/250
40/40 [==============================] - 0s 3ms/step - loss: 9.9877 - me
an_squared_error: 9.9877
Epoch 206/250
40/40 [==============================] - 0s 2ms/step - loss: 10.2084 - m
ean_squared_error: 10.2084
Epoch 207/250
40/40 [==============================] - 0s 2ms/step - loss: 10.1571 - m
ean_squared_error: 10.1571
Epoch 208/250
40/40 [==============================] - 0s 2ms/step - loss: 10.0451 - m
ean_squared_error: 10.0451
Epoch 209/250
40/40 [==============================] - 0s 2ms/step - loss: 9.8389 - me
an_squared_error: 9.8389
Epoch 210/250
40/40 [==============================] - 0s 3ms/step - loss: 9.9054 - me
an_squared_error: 9.9054
Epoch 211/250
40/40 [==============================] - 0s 3ms/step - loss: 9.8044 - me
an_squared_error: 9.8044
Epoch 212/250
40/40 [==============================] - 0s 2ms/step - loss: 10.1872 - m
ean_squared_error: 10.1872
Epoch 213/250
40/40 [==============================] - 0s 2ms/step - loss: 10.3235 - m
ean_squared_error: 10.3235
Epoch 214/250
40/40 [==============================] - 0s 2ms/step - loss: 10.0446 - m
ean_squared_error: 10.0446
Epoch 215/250
40/40 [==============================] - 0s 3ms/step - loss: 10.1395 - m
ean_squared_error: 10.1395
Epoch 216/250
40/40 [==============================] - 0s 2ms/step - loss: 9.9383 - me
```

```
an_squared_error: 9.9383
Epoch 217/250
40/40 [==============================] - 0s 2ms/step - loss: 9.9736 - me
an_squared_error: 9.9736
Epoch 218/250
40/40 [==============================] - 0s 3ms/step - loss: 10.1609 - m
ean_squared_error: 10.1609
Epoch 219/250
40/40 [==============================] - 0s 2ms/step - loss: 10.2898 - m
ean_squared_error: 10.2898
Epoch 220/250
40/40 [==============================] - 0s 3ms/step - loss: 9.8462 - me
an_squared_error: 9.8462
Epoch 221/250
40/40 [==============================] - 0s 2ms/step - loss: 9.8679 - me
an_squared_error: 9.8679
Epoch 222/250
40/40 [==============================] - 0s 3ms/step - loss: 10.1119 - m
ean_squared_error: 10.1119
Epoch 223/250
40/40 [==============================] - 0s 2ms/step - loss: 9.9688 - me
an_squared_error: 9.9688
Epoch 224/250
40/40 [==============================] - 0s 3ms/step - loss: 9.7316 - me
an_squared_error: 9.7316
Epoch 225/250
40/40 [==============================] - 0s 3ms/step - loss: 9.8160 - me
an_squared_error: 9.8160
Epoch 226/250
40/40 [==============================] - 0s 3ms/step - loss: 10.0192 - m
ean_squared_error: 10.0192
Epoch 227/250
40/40 [==============================] - 0s 2ms/step - loss: 9.9912 - me
an_squared_error: 9.9912
Epoch 228/250
40/40 [==============================] - 0s 2ms/step - loss: 9.5271 - me
an_squared_error: 9.5271
Epoch 229/250
40/40 [==============================] - 0s 2ms/step - loss: 9.7226 - me
an_squared_error: 9.7226
Epoch 230/250
40/40 [==============================] - 0s 3ms/step - loss: 9.9019 - me
an_squared_error: 9.9019
Epoch 231/250
40/40 [==============================] - 0s 2ms/step - loss: 9.9981 - me
an_squared_error: 9.9981
Epoch 232/250
40/40 [==============================] - 0s 3ms/step - loss: 9.6025 - me
an_squared_error: 9.6025
Epoch 233/250
40/40 [==============================] - 0s 3ms/step - loss: 9.8780 - me
an_squared_error: 9.8780
Epoch 234/250
40/40 [==============================] - 0s 3ms/step - loss: 9.9243 - me
an_squared_error: 9.9243
Epoch 235/250
40/40 [==============================] - 0s 2ms/step - loss: 9.9666 - me
```

```
an_squared_error: 9.9666
Epoch 236/250
40/40 [==============================] - 0s 2ms/step - loss: 9.8163 - me
an_squared_error: 9.8163
Epoch 237/250
40/40 [==============================] - 0s 3ms/step - loss: 10.1646 - m
ean_squared_error: 10.1646
Epoch 238/250
40/40 [==============================] - 0s 3ms/step - loss: 9.8728 - me
an_squared_error: 9.8728
Epoch 239/250
40/40 [==============================] - 0s 3ms/step - loss: 9.6658 - me
an_squared_error: 9.6658
Epoch 240/250
40/40 [==============================] - 0s 2ms/step - loss: 10.0512 - m
ean_squared_error: 10.0512
Epoch 241/250
40/40 [==============================] - 0s 2ms/step - loss: 9.9027 - me
an_squared_error: 9.9027
Epoch 242/250
40/40 [==============================] - 0s 2ms/step - loss: 9.7588 - me
an_squared_error: 9.7588
Epoch 243/250
40/40 [==============================] - 0s 2ms/step - loss: 9.8366 - me
an_squared_error: 9.8366
Epoch 244/250
40/40 [==============================] - 0s 2ms/step - loss: 9.7600 - me
an_squared_error: 9.7600
Epoch 245/250
40/40 [==============================] - 0s 2ms/step - loss: 9.8540 - me
an_squared_error: 9.8540
Epoch 246/250
40/40 [==============================] - 0s 2ms/step - loss: 9.6471 - me
an_squared_error: 9.6471
Epoch 247/250
40/40 [==============================] - 0s 2ms/step - loss: 9.8105 - me
an_squared_error: 9.8105
Epoch 248/250
40/40 [==============================] - 0s 2ms/step - loss: 9.7220 - me
an_squared_error: 9.7220
Epoch 249/250
40/40 [==============================] - 0s 3ms/step - loss: 9.9400 - me
an_squared_error: 9.9400
Epoch 250/250
40/40 [==============================] - 0s 2ms/step - loss: 9.5385 - me
an_squared_error: 9.5385
```

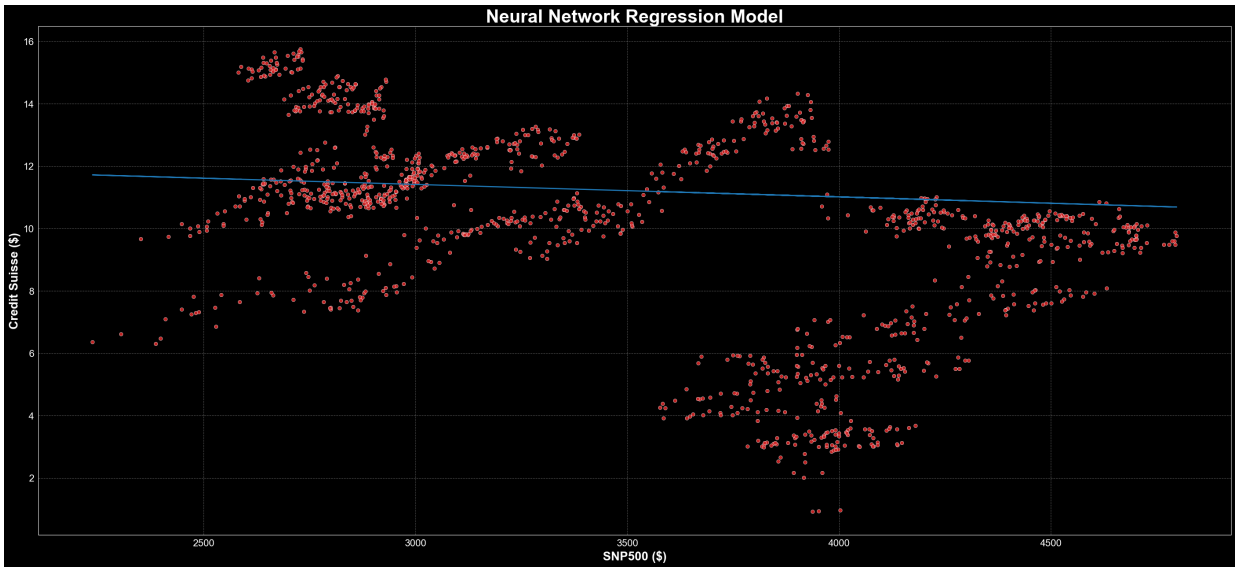
Out[148]: <keras.callbacks.History at 0x7fc8ba683e80>

In [149…

```
# Use the trained model to make predictions for the price of Credit Suiss
pred_y = model.predict(x)
```

```
40/40 [==============================] - 0s 1ms/step
```

In [150…
```
plt.style.use('dark_background')
plt.figure(figsize=(35,15))
sns.scatterplot(x=x.flatten(), y=y.flatten(), color='tab:red', alpha=0.9)
plt.plot(x, pred_y, color='tab:blue', linewidth=2)
plt.title('Neural Network Regression Model', fontsize=30, color="White",
plt.xlabel("SNP500 ($)", fontsize=20, fontweight='bold')
plt.ylabel("Credit Suisse ($)", fontsize=20, fontweight='bold')
plt.tick_params(axis="both", labelsize=16)
plt.grid(linestyle='--', color='gray', alpha=0.7)
plt.show()
```



Neural Network shows slightly little downward correlation between SNP500 and Credit suisse. neural network model is not the best way to determine if there is a correlation between two variables. A better way to do this is through a correlation analysis or linear regression analysis.

CREDIT SUISSE & FTSE100

In [151…
```
credit_suisse = yf.Ticker("CS").history(period="5y")
ftse100 = yf.Ticker("^FTSE").history(period="5y")
```

In [152…
```
from forex_python.converter import CurrencyRates
```

In [153…
```
c = CurrencyRates()
exchange_rate = c.get_rate('USD', 'GBP')
credit_suisse["Close"] = credit_suisse["Close"] * exchange_rate
credit_suisse["Open"] = credit_suisse["Open"] * exchange_rate
credit_suisse["High"] = credit_suisse["High"] * exchange_rate
credit_suisse["Low"] = credit_suisse["Low"] * exchange_rate
```

In [154…
```
# Preprocess the data by extracting the "Close" prices for Credit Suisse
y = credit_suisse["Close"].values
x = ftse100["Close"].values.reshape(-1, 1)
```

In [155…
```
y.shape
```

Out[155]: (1258,)

In [156…
```
x.shape
```

Out[156]: (1262, 1)

In [157…
```
x = x[4:]
x.shape
```

Out[157]: (1258, 1)

In [158…
```
from keras.layers import Dense, Dropout
```

In [159…
```
# Initialize the neural network model
model = Sequential()
model.add(Dense(64, input_shape=(1,), activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(1, activation='linear'))
```

In [160…
```
# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')
# Train the model
model.fit(x, y, epochs=250, batch_size=32, verbose=1)
```

```
Epoch 1/250
40/40 [==============================] - 1s 2ms/step - loss: 112108.1875
Epoch 2/250
40/40 [==============================] - 0s 2ms/step - loss: 24145.7812
Epoch 3/250
40/40 [==============================] - 0s 2ms/step - loss: 12658.5410
Epoch 4/250
40/40 [==============================] - 0s 2ms/step - loss: 9162.3779
Epoch 5/250
40/40 [==============================] - 0s 2ms/step - loss: 6844.7734
Epoch 6/250
40/40 [==============================] - 0s 2ms/step - loss: 5425.7324
Epoch 7/250
40/40 [==============================] - 0s 2ms/step - loss: 3857.2588
Epoch 8/250
40/40 [==============================] - 0s 2ms/step - loss: 2906.1746
Epoch 9/250
40/40 [==============================] - 0s 2ms/step - loss: 2298.6489
Epoch 10/250
40/40 [==============================] - 0s 2ms/step - loss: 1898.8218
Epoch 11/250
40/40 [==============================] - 0s 2ms/step - loss: 1353.1500
Epoch 12/250
40/40 [==============================] - 0s 2ms/step - loss: 933.7247
Epoch 13/250
40/40 [==============================] - 0s 3ms/step - loss: 783.3883
Epoch 14/250
40/40 [==============================] - 0s 2ms/step - loss: 600.6353
```

```
Epoch 15/250
40/40 [==============================] - 0s 2ms/step - loss: 516.9771
Epoch 16/250
40/40 [==============================] - 0s 2ms/step - loss: 355.8154
Epoch 17/250
40/40 [==============================] - 0s 3ms/step - loss: 297.3196
Epoch 18/250
40/40 [==============================] - 0s 2ms/step - loss: 260.0709
Epoch 19/250
40/40 [==============================] - 0s 2ms/step - loss: 230.9619
Epoch 20/250
40/40 [==============================] - 0s 2ms/step - loss: 199.2812
Epoch 21/250
40/40 [==============================] - 0s 2ms/step - loss: 157.5069
Epoch 22/250
40/40 [==============================] - 0s 3ms/step - loss: 133.6286
Epoch 23/250
40/40 [==============================] - 0s 2ms/step - loss: 128.6620
Epoch 24/250
40/40 [==============================] - 0s 2ms/step - loss: 91.5371
Epoch 25/250
40/40 [==============================] - 0s 2ms/step - loss: 93.5894
Epoch 26/250
40/40 [==============================] - 0s 3ms/step - loss: 80.8802
Epoch 27/250
40/40 [==============================] - 0s 2ms/step - loss: 62.4284
Epoch 28/250
40/40 [==============================] - 0s 3ms/step - loss: 63.1368
Epoch 29/250
40/40 [==============================] - 0s 2ms/step - loss: 59.8204
Epoch 30/250
40/40 [==============================] - 0s 2ms/step - loss: 50.1905
Epoch 31/250
40/40 [==============================] - 0s 2ms/step - loss: 53.0755
Epoch 32/250
40/40 [==============================] - 0s 2ms/step - loss: 47.4847
Epoch 33/250
40/40 [==============================] - 0s 2ms/step - loss: 37.1428
Epoch 34/250
40/40 [==============================] - 0s 2ms/step - loss: 30.9005
Epoch 35/250
40/40 [==============================] - 0s 2ms/step - loss: 34.6390
Epoch 36/250
40/40 [==============================] - 0s 2ms/step - loss: 33.7869
Epoch 37/250
40/40 [==============================] - 0s 2ms/step - loss: 30.8541
Epoch 38/250
40/40 [==============================] - 0s 2ms/step - loss: 25.9107
Epoch 39/250
40/40 [==============================] - 0s 2ms/step - loss: 28.3282
Epoch 40/250
40/40 [==============================] - 0s 2ms/step - loss: 31.3064
Epoch 41/250
40/40 [==============================] - 0s 2ms/step - loss: 27.9882
Epoch 42/250
40/40 [==============================] - 0s 2ms/step - loss: 30.1238
Epoch 43/250
```

```
40/40 [==============================] - 0s 2ms/step - loss: 32.7356
Epoch 44/250
40/40 [==============================] - 0s 2ms/step - loss: 26.4706
Epoch 45/250
40/40 [==============================] - 0s 2ms/step - loss: 29.0070
Epoch 46/250
40/40 [==============================] - 0s 2ms/step - loss: 28.5526
Epoch 47/250
40/40 [==============================] - 0s 3ms/step - loss: 28.1488
Epoch 48/250
40/40 [==============================] - 0s 2ms/step - loss: 22.5964
Epoch 49/250
40/40 [==============================] - 0s 2ms/step - loss: 24.3917
Epoch 50/250
40/40 [==============================] - 0s 2ms/step - loss: 26.4382
Epoch 51/250
40/40 [==============================] - 0s 3ms/step - loss: 21.5413
Epoch 52/250
40/40 [==============================] - 0s 2ms/step - loss: 33.2938
Epoch 53/250
40/40 [==============================] - 0s 2ms/step - loss: 24.6699
Epoch 54/250
40/40 [==============================] - 0s 2ms/step - loss: 25.8763
Epoch 55/250
40/40 [==============================] - 0s 4ms/step - loss: 24.3169
Epoch 56/250
40/40 [==============================] - 0s 3ms/step - loss: 18.4116
Epoch 57/250
40/40 [==============================] - 0s 2ms/step - loss: 29.6184
Epoch 58/250
40/40 [==============================] - 0s 2ms/step - loss: 21.4903
Epoch 59/250
40/40 [==============================] - 0s 2ms/step - loss: 18.3270
Epoch 60/250
40/40 [==============================] - 0s 2ms/step - loss: 24.7928
Epoch 61/250
40/40 [==============================] - 0s 3ms/step - loss: 26.9117
Epoch 62/250
40/40 [==============================] - 0s 3ms/step - loss: 14.1537
Epoch 63/250
40/40 [==============================] - 0s 2ms/step - loss: 15.1278
Epoch 64/250
40/40 [==============================] - 0s 2ms/step - loss: 14.5849
Epoch 65/250
40/40 [==============================] - 0s 2ms/step - loss: 22.0244
Epoch 66/250
40/40 [==============================] - 0s 2ms/step - loss: 17.1629
Epoch 67/250
40/40 [==============================] - 0s 2ms/step - loss: 18.8494
Epoch 68/250
40/40 [==============================] - 0s 2ms/step - loss: 21.8502
Epoch 69/250
40/40 [==============================] - 0s 2ms/step - loss: 14.7913
Epoch 70/250
40/40 [==============================] - 0s 2ms/step - loss: 17.7023
Epoch 71/250
40/40 [==============================] - 0s 2ms/step - loss: 19.3665
```

```
Epoch 72/250
40/40 [==============================] - 0s 3ms/step - loss: 15.8987
Epoch 73/250
40/40 [==============================] - 0s 2ms/step - loss: 15.0903
Epoch 74/250
40/40 [==============================] - 0s 2ms/step - loss: 20.0167
Epoch 75/250
40/40 [==============================] - 0s 2ms/step - loss: 14.8186
Epoch 76/250
40/40 [==============================] - 0s 2ms/step - loss: 24.2631
Epoch 77/250
40/40 [==============================] - 0s 2ms/step - loss: 24.3613
Epoch 78/250
40/40 [==============================] - 0s 2ms/step - loss: 19.6638
Epoch 79/250
40/40 [==============================] - 0s 3ms/step - loss: 16.7904
Epoch 80/250
40/40 [==============================] - 0s 4ms/step - loss: 14.7545
Epoch 81/250
40/40 [==============================] - 0s 2ms/step - loss: 20.6851
Epoch 82/250
40/40 [==============================] - 0s 2ms/step - loss: 17.8511
Epoch 83/250
40/40 [==============================] - 0s 3ms/step - loss: 17.4584
Epoch 84/250
40/40 [==============================] - 0s 2ms/step - loss: 19.4125
Epoch 85/250
40/40 [==============================] - 0s 2ms/step - loss: 14.7660
Epoch 86/250
40/40 [==============================] - 0s 2ms/step - loss: 12.9025
Epoch 87/250
40/40 [==============================] - 0s 2ms/step - loss: 18.9702
Epoch 88/250
40/40 [==============================] - 0s 2ms/step - loss: 16.7888
Epoch 89/250
40/40 [==============================] - 0s 2ms/step - loss: 13.6761
Epoch 90/250
40/40 [==============================] - 0s 2ms/step - loss: 12.5549
Epoch 91/250
40/40 [==============================] - 0s 2ms/step - loss: 12.3058
Epoch 92/250
40/40 [==============================] - 0s 2ms/step - loss: 14.9316
Epoch 93/250
40/40 [==============================] - 0s 3ms/step - loss: 15.9552
Epoch 94/250
40/40 [==============================] - 0s 2ms/step - loss: 17.2473
Epoch 95/250
40/40 [==============================] - 0s 2ms/step - loss: 12.7825
Epoch 96/250
40/40 [==============================] - 0s 2ms/step - loss: 14.9940
Epoch 97/250
40/40 [==============================] - 0s 3ms/step - loss: 26.8107
Epoch 98/250
40/40 [==============================] - 0s 2ms/step - loss: 12.4374
Epoch 99/250
40/40 [==============================] - 0s 2ms/step - loss: 15.2540
Epoch 100/250
```

```
40/40 [==============================] - 0s 2ms/step - loss: 13.9344
Epoch 101/250
40/40 [==============================] - 0s 2ms/step - loss: 16.1804
Epoch 102/250
40/40 [==============================] - 0s 2ms/step - loss: 12.5823
Epoch 103/250
40/40 [==============================] - 0s 2ms/step - loss: 16.2969
Epoch 104/250
40/40 [==============================] - 0s 3ms/step - loss: 11.5930
Epoch 105/250
40/40 [==============================] - 0s 2ms/step - loss: 12.1515
Epoch 106/250
40/40 [==============================] - 0s 2ms/step - loss: 14.5696
Epoch 107/250
40/40 [==============================] - 0s 2ms/step - loss: 11.0423
Epoch 108/250
40/40 [==============================] - 0s 2ms/step - loss: 13.1002
Epoch 109/250
40/40 [==============================] - 0s 2ms/step - loss: 13.1287
Epoch 110/250
40/40 [==============================] - 0s 2ms/step - loss: 13.7187
Epoch 111/250
40/40 [==============================] - 0s 2ms/step - loss: 11.1524
Epoch 112/250
40/40 [==============================] - 0s 2ms/step - loss: 11.6636
Epoch 113/250
40/40 [==============================] - 0s 3ms/step - loss: 12.0611
Epoch 114/250
40/40 [==============================] - 0s 2ms/step - loss: 12.6403
Epoch 115/250
40/40 [==============================] - 0s 3ms/step - loss: 12.3370
Epoch 116/250
40/40 [==============================] - 0s 2ms/step - loss: 12.7986
Epoch 117/250
40/40 [==============================] - 0s 2ms/step - loss: 14.8276
Epoch 118/250
40/40 [==============================] - 0s 2ms/step - loss: 15.6242
Epoch 119/250
40/40 [==============================] - 0s 2ms/step - loss: 9.9763
Epoch 120/250
40/40 [==============================] - 0s 2ms/step - loss: 12.8535
Epoch 121/250
40/40 [==============================] - 0s 2ms/step - loss: 12.9085
Epoch 122/250
40/40 [==============================] - 0s 2ms/step - loss: 10.4671
Epoch 123/250
40/40 [==============================] - 0s 4ms/step - loss: 12.2869
Epoch 124/250
40/40 [==============================] - 0s 2ms/step - loss: 11.7275
Epoch 125/250
40/40 [==============================] - 0s 3ms/step - loss: 11.6791
Epoch 126/250
40/40 [==============================] - 0s 2ms/step - loss: 12.6186
Epoch 127/250
40/40 [==============================] - 0s 2ms/step - loss: 12.5605
Epoch 128/250
40/40 [==============================] - 0s 3ms/step - loss: 11.6494
```

```
Epoch 129/250
40/40 [==============================] - 0s 2ms/step - loss: 15.8437
Epoch 130/250
40/40 [==============================] - 0s 2ms/step - loss: 11.5068
Epoch 131/250
40/40 [==============================] - 0s 2ms/step - loss: 11.3558
Epoch 132/250
40/40 [==============================] - 0s 3ms/step - loss: 10.4703
Epoch 133/250
40/40 [==============================] - 0s 2ms/step - loss: 10.3163
Epoch 134/250
40/40 [==============================] - 0s 2ms/step - loss: 10.3389
Epoch 135/250
40/40 [==============================] - 0s 2ms/step - loss: 11.0496
Epoch 136/250
40/40 [==============================] - 0s 2ms/step - loss: 10.7497
Epoch 137/250
40/40 [==============================] - 0s 2ms/step - loss: 11.0689
Epoch 138/250
40/40 [==============================] - 0s 3ms/step - loss: 10.9875
Epoch 139/250
40/40 [==============================] - 0s 3ms/step - loss: 11.6130
Epoch 140/250
40/40 [==============================] - 0s 2ms/step - loss: 10.8864
Epoch 141/250
40/40 [==============================] - 0s 2ms/step - loss: 11.9314
Epoch 142/250
40/40 [==============================] - 0s 2ms/step - loss: 12.4327
Epoch 143/250
40/40 [==============================] - 0s 3ms/step - loss: 14.6726
Epoch 144/250
40/40 [==============================] - 0s 2ms/step - loss: 12.5369
Epoch 145/250
40/40 [==============================] - 0s 2ms/step - loss: 10.1776
Epoch 146/250
40/40 [==============================] - 0s 2ms/step - loss: 10.1378
Epoch 147/250
40/40 [==============================] - 0s 3ms/step - loss: 10.6622
Epoch 148/250
40/40 [==============================] - 0s 2ms/step - loss: 10.9564
Epoch 149/250
40/40 [==============================] - 0s 2ms/step - loss: 11.7159
Epoch 150/250
40/40 [==============================] - 0s 2ms/step - loss: 10.5266
Epoch 151/250
40/40 [==============================] - 0s 3ms/step - loss: 13.2002
Epoch 152/250
40/40 [==============================] - 0s 2ms/step - loss: 10.6198
Epoch 153/250
40/40 [==============================] - 0s 3ms/step - loss: 11.9776
Epoch 154/250
40/40 [==============================] - 0s 2ms/step - loss: 10.4010
Epoch 155/250
40/40 [==============================] - 0s 2ms/step - loss: 10.6400
Epoch 156/250
40/40 [==============================] - 0s 3ms/step - loss: 10.0758
Epoch 157/250
```

```
40/40 [==============================] - 0s 2ms/step - loss: 11.0498
Epoch 158/250
40/40 [==============================] - 0s 2ms/step - loss: 11.1585
Epoch 159/250
40/40 [==============================] - 0s 2ms/step - loss: 9.7133
Epoch 160/250
40/40 [==============================] - 0s 2ms/step - loss: 10.2849
Epoch 161/250
40/40 [==============================] - 0s 3ms/step - loss: 10.9232
Epoch 162/250
40/40 [==============================] - 0s 2ms/step - loss: 10.5189
Epoch 163/250
40/40 [==============================] - 0s 3ms/step - loss: 10.1461
Epoch 164/250
40/40 [==============================] - 0s 2ms/step - loss: 12.1645
Epoch 165/250
40/40 [==============================] - 0s 2ms/step - loss: 10.8810
Epoch 166/250
40/40 [==============================] - 0s 3ms/step - loss: 9.9378
Epoch 167/250
40/40 [==============================] - 0s 2ms/step - loss: 14.3815
Epoch 168/250
40/40 [==============================] - 0s 2ms/step - loss: 11.2885
Epoch 169/250
40/40 [==============================] - 0s 4ms/step - loss: 10.8374
Epoch 170/250
40/40 [==============================] - 0s 4ms/step - loss: 10.7823
Epoch 171/250
40/40 [==============================] - 0s 3ms/step - loss: 10.1916
Epoch 172/250
40/40 [==============================] - 0s 3ms/step - loss: 10.9079
Epoch 173/250
40/40 [==============================] - 0s 2ms/step - loss: 10.6876
Epoch 174/250
40/40 [==============================] - 0s 2ms/step - loss: 9.9753
Epoch 175/250
40/40 [==============================] - 0s 2ms/step - loss: 11.1332
Epoch 176/250
40/40 [==============================] - 0s 2ms/step - loss: 9.6604
Epoch 177/250
40/40 [==============================] - 0s 3ms/step - loss: 10.3471
Epoch 178/250
40/40 [==============================] - 0s 2ms/step - loss: 10.0717
Epoch 179/250
40/40 [==============================] - 0s 3ms/step - loss: 9.7601
Epoch 180/250
40/40 [==============================] - 0s 2ms/step - loss: 9.9894
Epoch 181/250
40/40 [==============================] - 0s 2ms/step - loss: 14.6059
Epoch 182/250
40/40 [==============================] - 0s 2ms/step - loss: 10.5480
Epoch 183/250
40/40 [==============================] - 0s 2ms/step - loss: 9.6687
Epoch 184/250
40/40 [==============================] - 0s 2ms/step - loss: 9.9771
Epoch 185/250
40/40 [==============================] - 0s 3ms/step - loss: 9.5606
```

```
Epoch 186/250
40/40 [==============================] - 0s 3ms/step - loss: 10.3088
Epoch 187/250
40/40 [==============================] - 0s 2ms/step - loss: 10.1681
Epoch 188/250
40/40 [==============================] - 0s 3ms/step - loss: 10.7547
Epoch 189/250
40/40 [==============================] - 0s 3ms/step - loss: 11.1517
Epoch 190/250
40/40 [==============================] - 0s 2ms/step - loss: 10.7906
Epoch 191/250
40/40 [==============================] - 0s 2ms/step - loss: 10.2115
Epoch 192/250
40/40 [==============================] - 0s 3ms/step - loss: 10.6327
Epoch 193/250
40/40 [==============================] - 0s 2ms/step - loss: 9.6791
Epoch 194/250
40/40 [==============================] - 0s 3ms/step - loss: 11.1618
Epoch 195/250
40/40 [==============================] - 0s 2ms/step - loss: 10.7405
Epoch 196/250
40/40 [==============================] - 0s 2ms/step - loss: 10.1283
Epoch 197/250
40/40 [==============================] - 0s 2ms/step - loss: 11.2806
Epoch 198/250
40/40 [==============================] - 0s 3ms/step - loss: 9.8216
Epoch 199/250
40/40 [==============================] - 0s 2ms/step - loss: 9.1689
Epoch 200/250
40/40 [==============================] - 0s 3ms/step - loss: 11.1139
Epoch 201/250
40/40 [==============================] - 0s 2ms/step - loss: 11.0455
Epoch 202/250
40/40 [==============================] - 0s 2ms/step - loss: 10.0862
Epoch 203/250
40/40 [==============================] - 0s 2ms/step - loss: 10.4113
Epoch 204/250
40/40 [==============================] - 0s 3ms/step - loss: 9.8511
Epoch 205/250
40/40 [==============================] - 0s 2ms/step - loss: 9.8214
Epoch 206/250
40/40 [==============================] - 0s 2ms/step - loss: 9.6006
Epoch 207/250
40/40 [==============================] - 0s 2ms/step - loss: 9.6381
Epoch 208/250
40/40 [==============================] - 0s 4ms/step - loss: 11.0725
Epoch 209/250
40/40 [==============================] - 0s 2ms/step - loss: 9.1599
Epoch 210/250
40/40 [==============================] - 0s 2ms/step - loss: 10.2337
Epoch 211/250
40/40 [==============================] - 0s 3ms/step - loss: 9.5471
Epoch 212/250
40/40 [==============================] - 0s 3ms/step - loss: 9.7818
Epoch 213/250
40/40 [==============================] - 0s 2ms/step - loss: 9.7196
Epoch 214/250
```

```
40/40 [==============================] - 0s 3ms/step - loss: 10.9514
Epoch 215/250
40/40 [==============================] - 0s 2ms/step - loss: 9.9404
Epoch 216/250
40/40 [==============================] - 0s 2ms/step - loss: 10.0737
Epoch 217/250
40/40 [==============================] - 0s 2ms/step - loss: 9.5830
Epoch 218/250
40/40 [==============================] - 0s 3ms/step - loss: 9.4898
Epoch 219/250
40/40 [==============================] - 0s 2ms/step - loss: 9.2166
Epoch 220/250
40/40 [==============================] - 0s 2ms/step - loss: 9.9661
Epoch 221/250
40/40 [==============================] - 0s 3ms/step - loss: 9.7176
Epoch 222/250
40/40 [==============================] - 0s 3ms/step - loss: 10.2947
Epoch 223/250
40/40 [==============================] - 0s 2ms/step - loss: 9.6951
Epoch 224/250
40/40 [==============================] - 0s 2ms/step - loss: 10.4189
Epoch 225/250
40/40 [==============================] - 0s 3ms/step - loss: 10.3410
Epoch 226/250
40/40 [==============================] - 0s 2ms/step - loss: 10.2860
Epoch 227/250
40/40 [==============================] - 0s 2ms/step - loss: 9.9784
Epoch 228/250
40/40 [==============================] - 0s 2ms/step - loss: 9.5284
Epoch 229/250
40/40 [==============================] - 0s 2ms/step - loss: 10.1875
Epoch 230/250
40/40 [==============================] - 0s 2ms/step - loss: 9.4595
Epoch 231/250
40/40 [==============================] - 0s 2ms/step - loss: 9.2250
Epoch 232/250
40/40 [==============================] - 0s 2ms/step - loss: 9.4933
Epoch 233/250
40/40 [==============================] - 0s 2ms/step - loss: 9.6681
Epoch 234/250
40/40 [==============================] - 0s 3ms/step - loss: 9.1024
Epoch 235/250
40/40 [==============================] - 0s 3ms/step - loss: 9.0779
Epoch 236/250
40/40 [==============================] - 0s 2ms/step - loss: 10.2677
Epoch 237/250
40/40 [==============================] - 0s 2ms/step - loss: 9.5208
Epoch 238/250
40/40 [==============================] - 0s 3ms/step - loss: 9.7747
Epoch 239/250
40/40 [==============================] - 0s 2ms/step - loss: 9.0801
Epoch 240/250
40/40 [==============================] - 0s 3ms/step - loss: 9.9712
Epoch 241/250
40/40 [==============================] - 0s 2ms/step - loss: 9.2375
Epoch 242/250
40/40 [==============================] - 0s 2ms/step - loss: 9.1576
```

```
Epoch 243/250
40/40 [==============================] - 0s 2ms/step - loss: 9.1194
Epoch 244/250
40/40 [==============================] - 0s 4ms/step - loss: 9.3661
Epoch 245/250
40/40 [==============================] - 0s 4ms/step - loss: 9.1392
Epoch 246/250
40/40 [==============================] - 0s 2ms/step - loss: 9.1513
Epoch 247/250
40/40 [==============================] - 0s 3ms/step - loss: 9.8605
Epoch 248/250
40/40 [==============================] - 0s 3ms/step - loss: 9.1269
Epoch 249/250
40/40 [==============================] - 0s 2ms/step - loss: 9.5310
Epoch 250/250
40/40 [==============================] - 0s 2ms/step - loss: 10.7612
```

Out[160]: <keras.callbacks.History at 0x7fc8ba279a00>

In [161…
```
# Use the trained model to make predictions for the price of Credit Suiss
pred_y = model.predict(x)
```

```
40/40 [==============================] - 0s 1ms/step
```

In [162…
```
plt.style.use('dark_background')
plt.figure(figsize=(35,15))
sns.scatterplot(x=x.flatten(), y=y.flatten(), color='tab:red', alpha=0.9)
plt.plot(x, pred_y, color='tab:blue', linewidth=2)
plt.title('Neural Network Regression Model', fontsize=30, color="White",
plt.xlabel("FTSE100 (£)", fontsize=20, fontweight='bold')
plt.ylabel("Credit Suisse (£)", fontsize=20, fontweight='bold')
plt.tick_params(axis="both", labelsize=16)
plt.grid(linestyle='--', color='gray', alpha=0.7)
plt.show()
```