Bitcoin Data Exploration

Using Bitcoin Data sets to visualize data and uncover its story.

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors
```

Started with a CSV from kaggle

In [2]:
```
dfbitcoin = pd.read_csv("bitcoin.csv")
```

In [3]:
```
dfbitcoin.head()
```

Out[3]:

	date	price	total_volume	market_cap	coin_name
0	2015-01-01 00:00:00.000	313.992	4.699936e+07	4.293958e+09	bitcoin
1	2015-01-02 00:00:00.000	314.446	3.885591e+07	4.301448e+09	bitcoin
2	2015-01-03 00:00:00.000	286.572	1.187789e+08	3.921358e+09	bitcoin
3	2015-01-04 00:00:00.000	260.936	2.055001e+08	3.571640e+09	bitcoin
4	2015-01-05 00:00:00.000	273.220	1.550381e+08	3.740880e+09	bitcoin

Just looking at the price at a particular time does not give you much context. Its important to look at the market cap and also trends in price over time

In [4]:
```
# Ensure 'date' column is in datetime format for better plotting
dfbitcoin['date'] = pd.to_datetime(dfbitcoin['date'])
```

In [5]:
```
plt.figure(figsize=(20, 12))

# Create a line plot without markers for a cleaner look
plt.plot(dfbitcoin['date'], dfbitcoin['price'], linestyle='-', marker='', color='#FF7F

# Add a title and labels for axes with increased font size
plt.title('Bitcoin Price', fontsize=48)
plt.xlabel('Date', fontsize=32)
plt.ylabel('Price in USD', fontsize=32)

# Rotate date labels for better readability and increase font size
plt.xticks(rotation=45, fontsize=32)
plt.yticks(fontsize=32)

# Add gridlines for better readability of the plot
plt.grid(True)

plt.show()
```

I see a general uptrend but I cannot be sure what caused the significant price movements. How has the price responded to halvings in the past? What kind of opportunities does this offer and how can one use data to speculate within the space?

I overlaid the price with the total volume over time to see how it had an impact on price. Volume can refer to both buying and selling of an asset so I couldn't necessarily correlate the two.

In [6]:
```
dfbitcoin.head()
```

Out[6]:

	date	price	total_volume	market_cap	coin_name
0	2015-01-01	313.992	4.699936e+07	4.293958e+09	bitcoin
1	2015-01-02	314.446	3.885591e+07	4.301448e+09	bitcoin
2	2015-01-03	286.572	1.187789e+08	3.921358e+09	bitcoin
3	2015-01-04	260.936	2.055001e+08	3.571640e+09	bitcoin
4	2015-01-05	273.220	1.550381e+08	3.740880e+09	bitcoin

In [7]:
```
dfbitcoin['date'] = pd.to_datetime(dfbitcoin['date'])
```

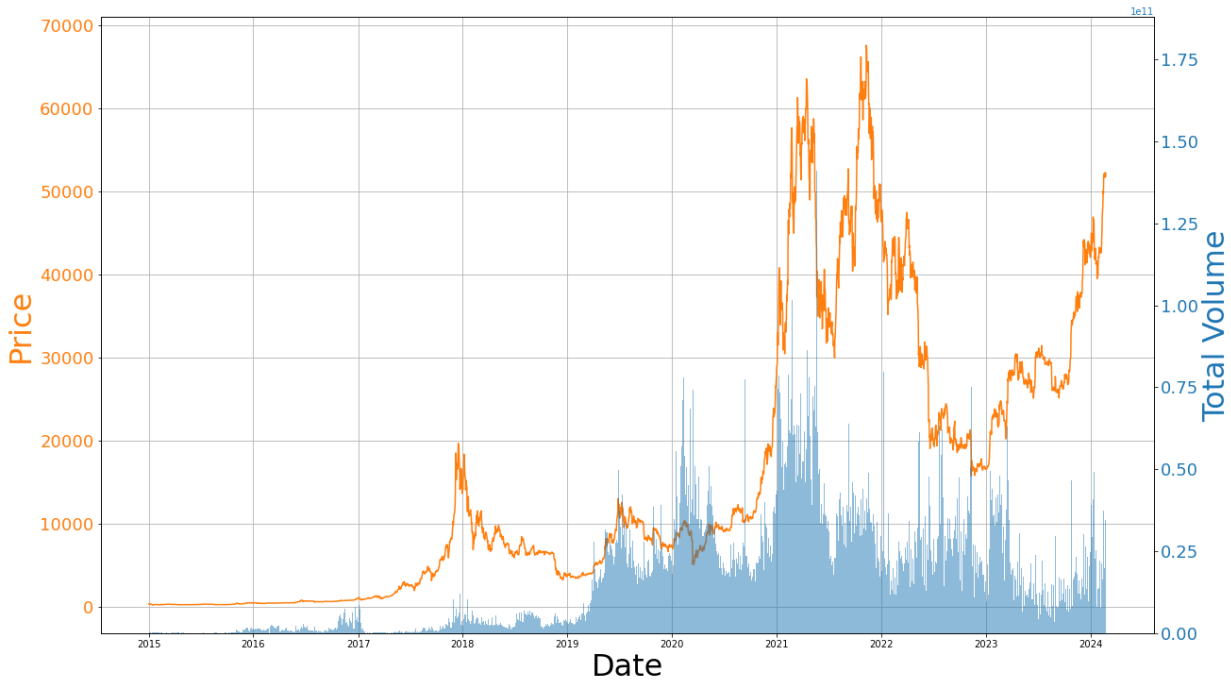
In [8]:
```
dfbitcoin.head()
```

Out[8]:

	date	price	total_volume	market_cap	coin_name
0	2015-01-01	313.992	4.699936e+07	4.293958e+09	bitcoin
1	2015-01-02	314.446	3.885591e+07	4.301448e+09	bitcoin
2	2015-01-03	286.572	1.187789e+08	3.921358e+09	bitcoin
3	2015-01-04	260.936	2.055001e+08	3.571640e+09	bitcoin
4	2015-01-05	273.220	1.550381e+08	3.740880e+09	bitcoin

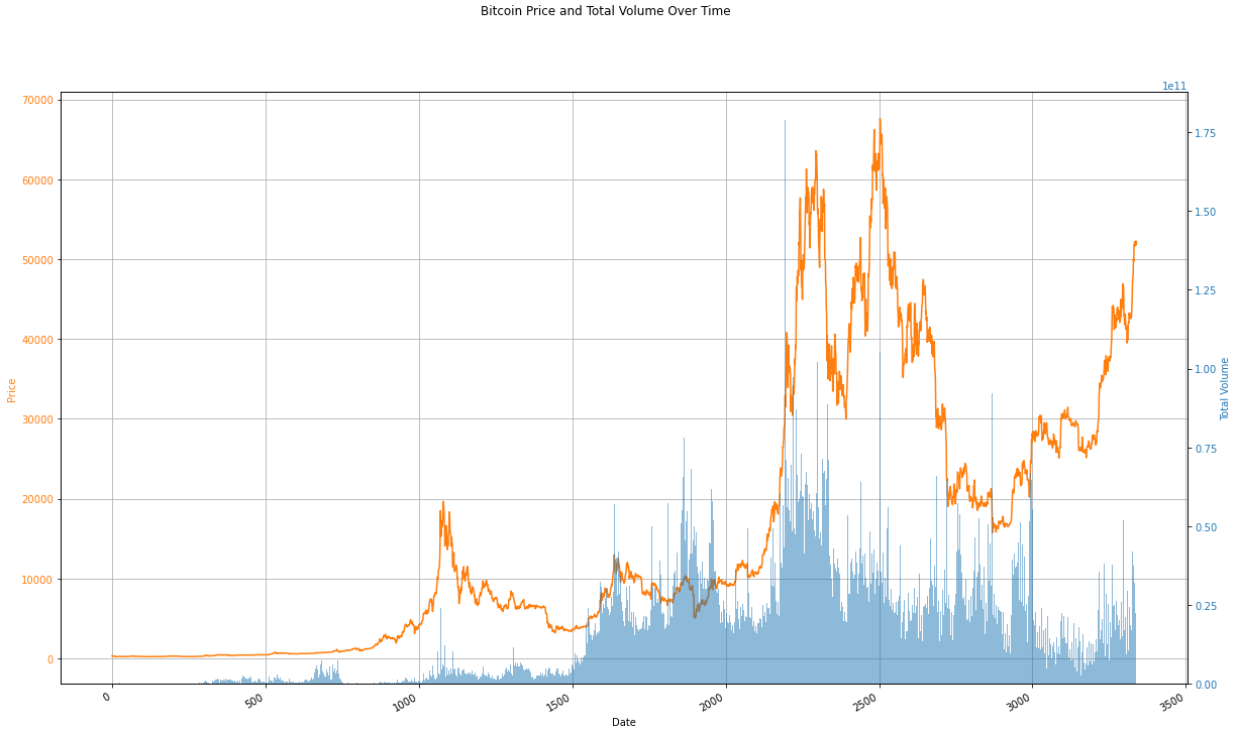
In [38]:

```
# Create a new figure and axis object
fig, ax1 = plt.subplots(figsize=(20, 12))
tableau_colors = list(mcolors.TABLEAU_COLORS)  # Gets a list of color names in TABLEAU
# Plot 'price' on the primary y-axis
color_price = tableau_colors[1]
ax1.set_xlabel('Date', fontsize = 32)
ax1.set_ylabel('Price', color=color_price, fontsize = 32)
ax1.plot(dfbitcoin.index, dfbitcoin['price'], color=color_price)
ax1.tick_params(axis='y', labelcolor=color_price, labelsize = 18)

# Instantiate a second y-axis sharing the same x-axis for the volume bar graph
ax2 = ax1.twinx()
color_volume = tableau_colors[0]
ax2.set_ylabel('Total Volume', color=color_volume, fontsize = 32)

# Because we're plotting bars, we need to specify the width of each bar so they're vis
time_range = (dfbitcoin.index.max() - dfbitcoin.index.min())
bar_width = time_range / len(dfbitcoin.index)

ax2.bar(dfbitcoin.index, dfbitcoin['total_volume'], width=bar_width, alpha=0.5, color=
ax2.tick_params(axis='y', labelcolor=color_volume, labelsize = 18)

# Set title and show grid
fig.suptitle('Bitcoin Price vs. Total Trading Volume', fontsize = 48)
ax1.grid(True)

# Rotate date labels for better readability
fig.autofmt_xdate

plt.savefig('BTC over volume.jpeg', format='jpeg', dpi=300)

# Show the plot
plt.show()
```



In [10]:
```
# Gets a list of color names in TABLEAU_COLORS
tableau_colors = list(mcolors.TABLEAU_COLORS.values())

# Create a new figure and axis object
fig, ax1 = plt.subplots(figsize=(20, 12))

# Plot 'price' on the primary y-axis
color_price = tableau_colors[1]
ax1.set_xlabel('Date')
ax1.set_ylabel('Price', color=color_price)
ax1.plot(dfbitcoin.index, dfbitcoin['price'], color=color_price)
ax1.tick_params(axis='y', labelcolor=color_price)

# Instantiate a second y-axis sharing the same x-axis for the volume bar graph
ax2 = ax1.twinx()
color_volume = tableau_colors[0]
ax2.set_ylabel('Total Volume', color=color_volume)

# Adjust the width of each bar so they're visible
# Here, we calculate the total number of days divided by the number of data points to
time_range = (dfbitcoin.index.max() - dfbitcoin.index.min())
bar_width = time_range / len(dfbitcoin.index)

ax2.bar(dfbitcoin.index, dfbitcoin['total_volume'], width=bar_width, alpha=0.5, color=
ax2.tick_params(axis='y', labelcolor=color_volume)

# Set title and show grid
fig.suptitle('Bitcoin Price and Total Volume Over Time')
ax1.grid(True)

# Rotate date labels for better readability
fig.autofmt_xdate()

#plt.savefig('BTC_over_volume.jpeg', format='jpeg', dpi=300)
```

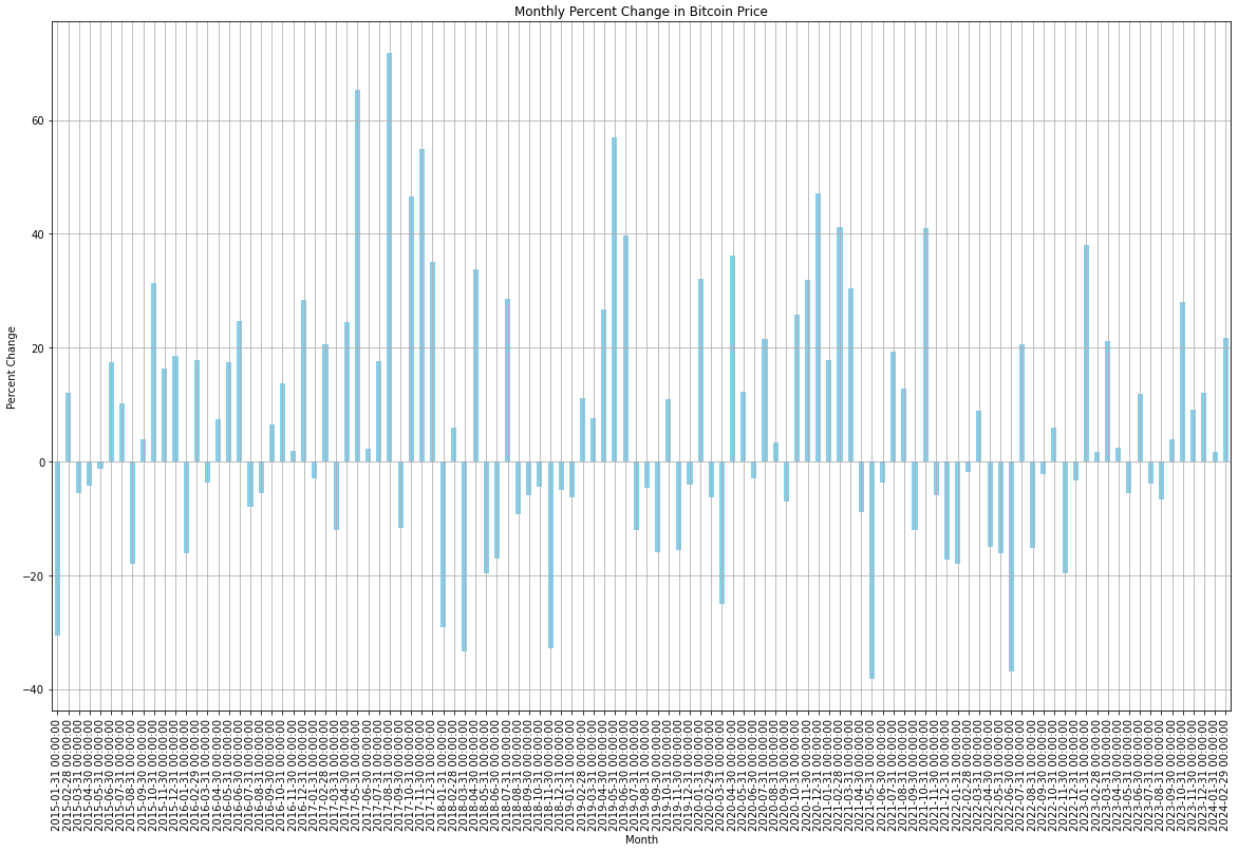
```
plt.show()
```



Although the price action looks very exciting as time goes on I wanted to know what the actual profit was for someone entering the market. I modified the data set to show the monthly percentage change of Bitcoin.

In [11]:
```
# Get the opening price of the month (first day of the month)
# Convert the 'date' column to datetime format
dfbitcoin['date'] = pd.to_datetime(dfbitcoin['date'])

# Set the 'date' column as the index of the DataFrame
dfbitcoin.set_index('date', inplace=True)
```

In [12]:
```
monthly_open = dfbitcoin.resample('MS').first()  # 'MS' stands for Month Start

# Get the closing price of the month (last day of the month)
monthly_close = dfbitcoin.resample('M').last()  # 'M' stands for Month End
monthly_open.index = monthly_close.index

# Calculate the percent change from the opening to the closing price of the month
btc_monthly_percent_change = ((monthly_close['price'] - monthly_open['price']) / month
btc_monthly_percent_change.to_csv('BTC_PERCENT_Change.csv', index=True)

# Plot the percent change
plt.figure(figsize=(20, 12))
btc_monthly_percent_change.plot(kind='bar', color='skyblue')
plt.title('Monthly Percent Change in Bitcoin Price')
plt.ylabel('Percent Change')
plt.xlabel('Month')
plt.grid(True)
plt.show()
```



Although the price action looks very exciting as time goes on I wanted to know what the actual profit was for someone entering the market. I modified the data set to show the monthly percentage change of Bitcoin.

Surprisingly it seems that the overall percentage change of Bitcoin was way higher when the pricer was lower and you can see diminishing returns as time moves forward. This was still pretty difficult to visualize how the asset performed so I wanted to see it grouped over a year to see the reward of long term holders.

In [13]:
```
# Get the opening price of the year (first day of the year)
yearly_open = dfbitcoin.resample('YS').first()['price']  # 'YS' stands for Year Start

# Get the closing price of the year (last day of the year)
yearly_close = dfbitcoin.resample('Y').last()['price']  # 'Y' stands for Year End

yearly_open.index = yearly_close.index

# Calculate the percent change from the opening to the closing price of the year
btc_yearly_percent_change = ((yearly_close - yearly_open) / yearly_open) * 100

# Save the yearly percent change to a CSV file
#btc_yearly_percent_change.to_csv('BTC_YEARLY_PERCENT_Change.csv', index=True)
```

In [14]:
```
print(btc_monthly_percent_change)
```

```
date
2015-01-31   -30.564059
2015-02-28    12.078241
2015-03-31    -5.602592
2015-04-30    -4.277045
2015-05-31    -1.315544
                 ...
2023-10-31    27.915693
2023-11-30     9.050620
2023-12-31    11.955915
2024-01-31     1.620141
2024-02-29    21.744442
Freq: M, Name: price, Length: 110, dtype: float64
```

In [15]:
```
dfEth = pd.read_csv("ethereum.csv")
dfEth['date'] = pd.to_datetime(dfEth['date'])
dfEth.set_index('date', inplace=True)
```

In [16]:
```
dfEth.head()
```

Out[16]:

date	price	total_volume	market_cap	coin_name
2015-08-07	2.831620	9.062200e+04	0.000000e+00	ethereum
2015-08-08	1.330750	3.680700e+05	8.033948e+07	ethereum
2015-08-10	0.687586	4.004641e+05	4.155631e+07	ethereum
2015-08-11	1.067379	1.518998e+06	6.453901e+07	ethereum
2015-08-12	1.256613	2.073893e+06	7.601326e+07	ethereum

In [17]:
```
dfbitcoin.head()
```

Out[17]:

date	price	total_volume	market_cap	coin_name
2015-01-01	313.992	4.699936e+07	4.293958e+09	bitcoin
2015-01-02	314.446	3.885591e+07	4.301448e+09	bitcoin
2015-01-03	286.572	1.187789e+08	3.921358e+09	bitcoin
2015-01-04	260.936	2.055001e+08	3.571640e+09	bitcoin
2015-01-05	273.220	1.550381e+08	3.740880e+09	bitcoin

In [18]:
```
combined_df = pd.concat([dfbitcoin, dfEth], axis=0)
```

In [19]:
```
# Save the combined DataFrame to a CSV file in the current working directory
#combined_df.to_csv('combined_crypto_data.csv', index=True)
```

As Bitcoin clearly offers attractive returns at a higher risk than the S&P 500 I wanted to see what other similar assets there were in the crypto space. I really liked how this visual came out because it shows Bitcoin in 2015 as a small emerging asset class. As it grows and other projects

enter the space, this visual shows how significant they are compared to the early days of Bitcoin. Of course they are much smaller but I wanted to explore these assets a little more.

It was really hard to compare these bars so I wanted to get a better visual of how these returns compared to Bitcoin. It confirmed that the returns were way bigger than Bitcoins. I know in class we talked about not using area to compare things. But in this case it really helps me visualize the data. I feel that many people would dismiss the performance of these other projects, but when viewed from this perspective it does show that they are worth paying attention to.

In [20]:
```
# Load additional DataFrames
dfChainlink = pd.read_csv("chainlink.csv")
dfAvalanche = pd.read_csv("avalanche-2.csv")
dfCardano = pd.read_csv("cardano.csv")
dfPolkadot = pd.read_csv("polkadot.csv")
dfVechain = pd.read_csv("vechain.csv")

# Convert 'date' to datetime and set as index for each DataFrame
data_frames = [dfChainlink, dfAvalanche, dfCardano, dfPolkadot, dfVechain]
for df in data_frames:
    df['date'] = pd.to_datetime(df['date'])
    df.set_index('date', inplace=True)

# Concatenate all DataFrames
combined_df = pd.concat([dfbitcoin, dfEth, dfChainlink, dfAvalanche, dfCardano, dfPolk

# Save the combined DataFrame to a CSV file in the current working directory
#combined_df.to_csv('combined_crypto_data.csv', index=True)
```

In [21]:
```
def calculate_monthly_percent_change(df):
    # Assuming 'date' is already the DataFrame's index, so no need to reset it here
    monthly_open = df.resample('MS').first()  # Month Start
    monthly_close = df.resample('M').last()   # Month End
    monthly_open.index = monthly_close.index
    monthly_percent_change = ((monthly_close['price'] - monthly_open['price']) / montl

    return monthly_percent_change


# Apply the function to each coin in the combined DataFrame
result_df = pd.DataFrame()
for coin in combined_df['coin_name'].unique():
    coin_df = combined_df[combined_df['coin_name'] == coin]
    percent_change = calculate_monthly_percent_change(coin_df)
    percent_change = percent_change.reset_index()  # Reset index to turn the date inde
    percent_change['coin_name'] = coin  # Add a column for coin name
    result_df = pd.concat([result_df, percent_change], axis=0)

# Reset index of the result DataFrame to get a clean CSV
result_df.reset_index(drop=True, inplace=True)

# Save to CSV
#result_df.to_csv('monthly_percent_change_by_coin.csv', index=False)
```

In [22]:
```
result_df
```

Out[22]:

	date	price	coin_name
0	2015-01-31	-30.564059	bitcoin
1	2015-02-28	12.078241	bitcoin
2	2015-03-31	-5.602592	bitcoin
3	2015-04-30	-4.277045	bitcoin
4	2015-05-31	-1.315544	bitcoin
...	...	...	...
514	2023-10-31	9.376491	vechain
515	2023-11-30	15.143645	vechain
516	2023-12-31	67.539916	vechain
517	2024-01-31	-16.442622	vechain
518	2024-02-29	53.754885	vechain

519 rows × 3 columns

In [23]:
```
sp500df = pd.read_csv("SP500.csv")
```

In [24]:
```
sp500df.head
```

Out[24]:
```
<bound method NDFrame.head of              date       open       high        low
close      volume  \
0      1927-12-30    17.6600    17.6600    17.6600    17.6600           0
1      1928-01-03    17.7600    17.7600    17.7600    17.7600           0
2      1928-01-04    17.7200    17.7200    17.7200    17.7200           0
3      1928-01-05    17.5500    17.5500    17.5500    17.5500           0
4      1928-01-06    17.6600    17.6600    17.6600    17.6600           0
...           ...        ...        ...        ...        ...         ...
24147  2024-02-16  5031.1299  5038.7002  4999.5200  5005.5698  3833270000
24148  2024-02-20  4989.3198  4993.7100  4955.0200  4975.5098  4034880000
24149  2024-02-21  4963.0298  4983.2100  4946.0000  4981.7998  3788390000
24150  2024-02-22  5038.8301  5094.3901  5038.8301  5087.0298  4051710000
24151  2024-02-23  5100.9199  5111.0601  5081.4600  5088.7998  3672790000

       change_percent   avg_vol_20d
0                 NaN           NaN
1                0.57           NaN
2               -0.23           NaN
3               -0.96           NaN
4                0.63           NaN
...               ...           ...
24147           -0.48  4.093593e+09
24148           -0.60  4.080457e+09
24149            0.13  4.074236e+09
24150            2.11  4.060320e+09
24151            0.03  4.042938e+09

[24152 rows x 8 columns]>
```

In [25]:
```
sp500df['date'] = pd.to_datetime(sp500df['date'])
```

```
# Step 2: Set 'date' as the index
sp500df.set_index('date', inplace=True)
```

In [26]:
```
# Step 3: Filter for dates after 2015
sp500df_filtered = sp500df[sp500df.index.year >= 2015]

# Step 4: Resample to get the monthly open (first value) and monthly close (last value
monthly_open = sp500df_filtered.resample('MS').first()['close']
monthly_close = sp500df_filtered.resample('M').last()['close']
monthly_open.index = monthly_close.index


# Step 5: Calculate the monthly percent change
sp500_monthly_percent_change = ((monthly_close - monthly_open) / monthly_open) * 100
```

In [27]:
```
sp500_monthly_percent_change
```

Out[27]:
```
date
2015-01-31   -3.071130
2015-02-28    4.139347
2015-03-31   -2.337784
2015-04-30    1.253592
2015-05-31   -0.042693
                ...
2023-10-31   -2.205730
2023-11-30    7.785531
2023-12-31    3.813152
2024-01-31    2.167900
2024-02-29    3.722031
Freq: M, Name: close, Length: 110, dtype: float64
```

In [28]:
```
btc_monthly_percent_change
```

Out[28]:
```
date
2015-01-31   -30.564059
2015-02-28    12.078241
2015-03-31    -5.602592
2015-04-30    -4.277045
2015-05-31    -1.315544
                 ...
2023-10-31    27.915693
2023-11-30     9.050620
2023-12-31    11.955915
2024-01-31     1.620141
2024-02-29    21.744442
Freq: M, Name: price, Length: 110, dtype: float64
```

In [29]:
```
# Step 1: Combine the Series into a DataFrame
combined_df = pd.DataFrame({
    'SP500_Percent_Change': sp500_monthly_percent_change,
    'BTC_Percent_Change': btc_monthly_percent_change
})

# Step 2: Export the DataFrame to a CSV file
#combined_df.to_csv('sp_500combined_monthly_percent_change.csv', index=True)
```

In [30]:
```
combined_df
```

Out[30]:

date	SP500_Percent_Change	BTC_Percent_Change
2015-01-31	-3.071130	-30.564059
2015-02-28	4.139347	12.078241
2015-03-31	-2.337784	-5.602592
2015-04-30	1.253592	-4.277045
2015-05-31	-0.042693	-1.315544
...	...	...
2023-10-31	-2.205730	27.915693
2023-11-30	7.785531	9.050620
2023-12-31	3.813152	11.955915
2024-01-31	2.167900	1.620141
2024-02-29	3.722031	21.744442

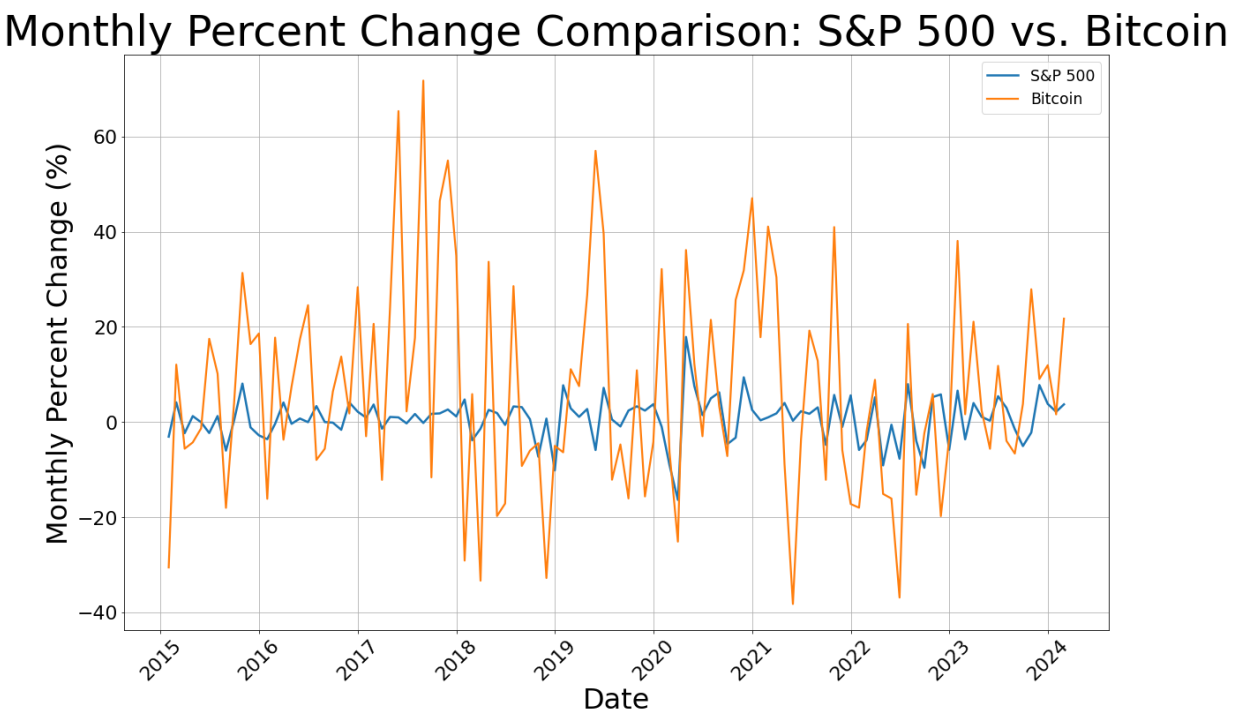
110 rows × 2 columns

As previously mentioned it seems that the asset had one of the biggest returns following the 2016 halving. Also the reversal following the 2016 halving brought more losses. It seems that the asset class not only diminishes rewards but also losses.

Seeing these numbers, I wanted to see how this compared to the S&P 500 so I found a separate csv file of the S&P500 and used python to merge these data sets so that I could plot them side by side.

In [55]:
```
plt.figure(figsize=(20, 12))  # Set the figure size for better readability

# Plot S&P 500 monthly percent change
plt.plot(combined_df.index, combined_df['SP500_Percent_Change'], label='S&P 500',linew

# Plot Bitcoin monthly percent change
plt.plot(combined_df.index, combined_df['BTC_Percent_Change'], label='Bitcoin',  lines

# Adding legend to distinguish the two lines
plt.legend(fontsize = 'xx-large')

# Adding title and labels
plt.title('Monthly Percent Change Comparison: S&P 500 vs. Bitcoin', fontsize = 48)
plt.ylabel('Monthly Percent Change (%)', fontsize = 32)
plt.xlabel('Date', fontsize=32)
# Show grid
plt.grid(True)


# Rotate date labels for better readability
plt.xticks(rotation=45, fontsize = 22)
plt.yticks(fontsize = 22)
```

```
plt.savefig('sp500_vs_bitcoin_monthly_change.jpeg', format='jpeg', dpi=300)

plt.show()
```



In [ ]: