Analysis of the Market Impact of the Federal Funds Rate

Mohammad Sheikhattari

INTRODUCTION

The objective of this project is to analyze federal funds rate data from 1954 to 2022, as well as stock price data in order to investigate the relationship between interest rates and stock prices.

The relationship between interest rates and stock prices is well-established in the financial community, since this rate underpins not only all commerce by determining the cost of money, but also valuations of financial instruments like stocks and bonds. Rising interest rates generally result in decreasing stock prices and vice versa, but the goal is to uncover nuances in this relationship. For example, is there any time delay between policy and stock valuations? How consistent is this relationship in varying time periods, each with their own unique set of macroeconomic circumstances? What kind of relationship do they share - is it linear? Finally, can we use this data to create a model for the current time period?

Imports

The below libraries were imported for the project.

In [11]:
```
import pandas as pd
import requests
from bs4 import BeautifulSoup
from datetime import datetime, timedelta
import yfinance as yf
import numpy as np
import re
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score
from scipy.optimize import curve_fit
from scipy.stats import pearsonr, spearmanr, chi2_contingency
```

1. DATA COLLECTION

Federal Funds Rate Data

In this phase of the data science pipeline, data is collected and then wrangled so that it has a clean, tidy format. Federal funds rate data is publically available going back to 1954 on the federal reserve website where a csv of the data was retrieved. Weekly averages of rates were downloaded between 1954 and the present date.

The csv contains dates, as well as the relevant interest rates during that time. For the purposes of our dataset, we will focus on the first column of rates, which represents the federal funds rate.

The steps that are taken in order to read this data in and clean it up are shown below.

In [12]:
```
ffr = pd.read_csv("FRB_H15.csv") #Read in downloaded csv

ffr = ffr.drop(range(5)) #Drop irrelevant rows
ffr = ffr.reset_index() #Index starting at 0 again

for col in ffr.columns:
    if(col != "Series Description" and col != "Federal funds effective rate"):
        del ffr[col]
#Drop every column except for the date and rates

ffr = ffr.rename(columns={"Series Description": "Date"}) #Give date column more descriptive name
ffr #display dataframe
```

Out[12]:

	Date	Federal funds effective rate
0	7/7/54	1
1	7/14/54	1.22
2	7/21/54	0.57
3	7/28/54	0.63
4	8/4/54	0.27
...	...	...
3566	11/9/22	3.83
3567	11/16/22	3.83
3568	11/23/22	3.83
3569	11/30/22	3.83
3570	12/7/22	3.83

3571 rows × 2 columns

The CSV which contains weekly interest rate averages from 1954 to present day was downloaded as described above. It was then read into a dataframe using the **pd.read_csv()** function and the dataframe was then manipulated. The first 5 rows gave descriptive information on the data contained in their columns so they were dropped and the indices were reset. All columns besides the week and federal funds rate were dropped, since only the federal funds rate is of interest to this analysis. Finally the first column was renamed to "Date" in order to be more descriptive.

The dataframe is then shown in order to display the results of this parsing and cleanup.

Stock Data

Stock price data is also necessary for this analysis. Yahoo finance has historical stock data available for any ticker, including SP500. For the purposes of this project, the S&P500 weekly average price will be used, since this index is considered a representative of the stock market's performance.

The yahoo finance API was used since it can provide historical data for any ticker that is immediately provided in a pandas dataframe. We check the ticker ^GSPC since this represents the S&P500.

In [13]:
```
ticker = yf.Ticker("^GSPC") #Yfinance API pulls info on SP500
sp = ticker.history(start= "1954-07-07", end= "2022-12-07", interval = "1wk") #Get weekly price history over specified date range

for col in sp.columns:
    if(col != "Close"):
        del sp[col]
sp = sp.reset_index()
#Get rid of all columns besides closing price and index by order rather than date
t = []

for index, row in sp.iterrows():
    t.append(sp.at[index, "Date"].date().strftime('%m/%d/%y'))
#Getting date out of datetime object and appending to an array

sp["Date"] = t
sp
#Changing date dolumn to hold dates, not datetime and also displaying dataframe
```

Out[13]:

	Date	Close
0	07/05/54	30.139999
1	07/12/54	30.059999
2	07/19/54	30.309999
3	07/26/54	30.879999
4	08/02/54	30.379999
...	...	...
3566	11/07/22	3992.929932
3567	11/14/22	3965.340088
3568	11/21/22	4026.120117
3569	11/28/22	4071.699951
3570	12/05/22	3933.919922

3571 rows × 2 columns

Using the yfinance API **ticker.history()** function, SP500 weekly prices were retrieved from the same time period as the federal funds rate data. This function returns a pandas dataframe, so it was easy to immediately begin cleaning up this data. The only field kept was the closing price of the SP500 on each of those days, and the indices of the dataframe were then reset since the date was originally used to index the dataframe, and we want the date to be a column of the dataframe instead. The date was then sanitized to have the exact same format as the federal funds rate data set in order to remain consistent. The dataframe was displayed.

Earnings data

Earnings growth has resulted in generally increasing stock prices over a long enough time frame. So, it would be a good idea to standardize the stock price against this factor in order to get a more accurate reading of a possible correlation between federal funds rate and stock price. This can be done by dividing the sp500 price by earnings in order to get a price/earnings ratio which is a reading of financial sentiment and health that is more robust to confounding factors of growth, and instead indicative of financial sentiment and the health of markets.

In [14]:
```
req = requests.get("https://www.in2013dollars.com/us-economy/s-p-500-earnings") #Pull html for sp500 earnings
soup = BeautifulSoup(req.content)
table = str(soup.find_all("table")[1])
er = pd.read_html(table)[0] #BS object created and dataframe created from relevant table
del er["Change ($)"]
del er["Month-over-month (%)"]
er = er.drop(range(2))
er = er.drop(range(821,1823))
er = er.reset_index()
del er["index"]
er
#Irrelevant columns and rows deleted, indexing reset, and dataframe displayed
```

Out[14]:

	Year	Month	S&P 500 EPS ($)
0	2022	9	189.88
1	2022	8	190.67
2	2022	7	191.47
3	2022	6	192.26
4	2022	5	194.14
...	...	...	...
814	1954	11	2.72
815	1954	10	2.68
816	1954	9	2.63
817	1954	8	2.63
818	1954	7	2.62

819 rows × 3 columns

The SP500 monthly earnings data was scraped from https://www.in2013dollars.com/us-economy/s-p-500-earnings. The data was then tidied so that rows with invalid data, or with dates irrelevant to this analysis were dropped. Columns were given more descriptive names, and irrelevant columns were dropped.

The earnings data is monthly and a few months of data are missing. Earnings are typically reported on a quarterly basis, so weekly earnings data is too granular to be readily available. This should be fine since stock prices which fell within the same month and year as the reported earnings can be reported as having those earnings. This will result in the earnings data being off by up to a month, but this shouldn't make a large difference when analyzing the data over a big enough timeframe. The earnings are for the trailing twelve months, so in order to get the earnings multiples the price can be divided by the annual earnings.

Now that the data has been collected, it's time to begin processing it so that it's ready for analysis.

2. DATA MANAGEMENT

A dataframe must be constructed with the interest rate and stock price data for coinciding dates. The dates don't match exactly, since the weekly cutoffs for the data providers is slightly different. But, the data appears to consistently be off by two days, and both datasets are complete. So for each entry in the **sp** dataframe, the entry in the **ffr** database can be found which has a date within 2 days of it and appended to the stock price dataframe.

This results in slightly inaccurate datapoints, as the federal funds rate data points have dates that are 2 days off, but the federal funds rate is not expected to fluctuate materially within this range of time. It's not ideal but given the structural limitations of the data it's the best that can be done. It's unlikely to affect the large timeframe patterns very much either.

In [15]:
```
def closest(date):
    for index, row in ffr.iterrows():
        if(abs(datetime.strptime(date, "%m/%d/%y") - datetime.strptime(ffr.at[index, "Date"], "%m/%d/%y")) == timedelta(days = 2)):
            return float(ffr.at[index, "Federal funds effective rate"])
    return -1
#This function iterates through the ffr dataframe and checks the time between the provided date and the date at the
#current index. Once a difference of 2 days is found, the rate at that index is returned. If none is found then -1 is
#returned. This finds the closest matching rate to the provided date.

r = []

for index, row in sp.iterrows():
     r.append(closest(sp.at[index, "Date"]))
#A rates array is created, and the sp dataframe is iterated through, appending the closest matching rate for each date

sp["Rate"] = r
sp
#A rate column is created using this array and the dataframe is displayed
```

Out[15]:

	Date	Close	Rate
0	07/05/54	30.139999	1.00
1	07/12/54	30.059999	1.22
2	07/19/54	30.309999	0.57
3	07/26/54	30.879999	0.63
4	08/02/54	30.379999	0.27
...	...	...	...
3566	11/07/22	3992.929932	3.83
3567	11/14/22	3965.340088	3.83
3568	11/21/22	4026.120117	3.83
3569	11/28/22	4071.699951	3.83
3570	12/05/22	3933.919922	3.83

3571 rows × 3 columns

The function **closest(date)** was defined and used to accomplish the task of matching data points. It works by checking the **ffr** dataframe for data that has dates that are within 2 days of the **date** parameter, and returning the interest rate at that date. This function was called for each date in the **sp** dataframe and stored in the array **r**. A new column "Rate" was then added to the sp dataframe containing all of the data from this array

Next the earnings multiples need to be calculated and added as a new column. A crude approximation must be made since monthly data is being combined with weekly data, and a few months of data is also missing. If the stock price and earnings fall in the same month and year, they are combined to create an earnings multiple corresponding to that date. For the few months of missing data, the most recent earnings will be used.

In [16]:
```
def earnings(date):
    m, y = re.match("(\d+)/\d+/(\d+)", date).groups()
    m = int(m)
    if(int(y) > 22):
        y = int("19" + y)
    else:
        y = int("20" + y)
        if(y == 2022 and m > 9):
            m = 9

    for index, row in er.iterrows():
        if(er.at[index, "Year"] == y and er.at[index, "Month"] == m):
            return float(er.at[index, "S&P 500 EPS ($)"])
#This function first pulls the month and year from the specified date in a format that aligns with the er df
#Then the er df is iterated through and when the matching year and month are found, the earnings at that date are returned

m = []

for index, row in sp.iterrows():
    m.append(sp.at[index, "Close"]/earnings(sp.at[index, "Date"]))
#The sp dataframe is iterated through and a PE ratio is appended to the m array using the price at each entry and the
#earnings function at that entry defined earlier

sp["PE Ratio"] = m
sp
#The column PE Ratio is appended to the sp dataframe and it is shown
```

Out[16]:

	Date	Close	Rate	PE Ratio
0	07/05/54	30.139999	1.00	11.503817
1	07/12/54	30.059999	1.22	11.473282
2	07/19/54	30.309999	0.57	11.568702
3	07/26/54	30.879999	0.63	11.786259
4	08/02/54	30.379999	0.27	11.551330
...	...	...	...	...
3566	11/07/22	3992.929932	3.83	21.028702
3567	11/14/22	3965.340088	3.83	20.883401
3568	11/21/22	4026.120117	3.83	21.203498
3569	11/28/22	4071.699951	3.83	21.443543
3570	12/05/22	3933.919922	3.83	20.717927

3571 rows × 4 columns

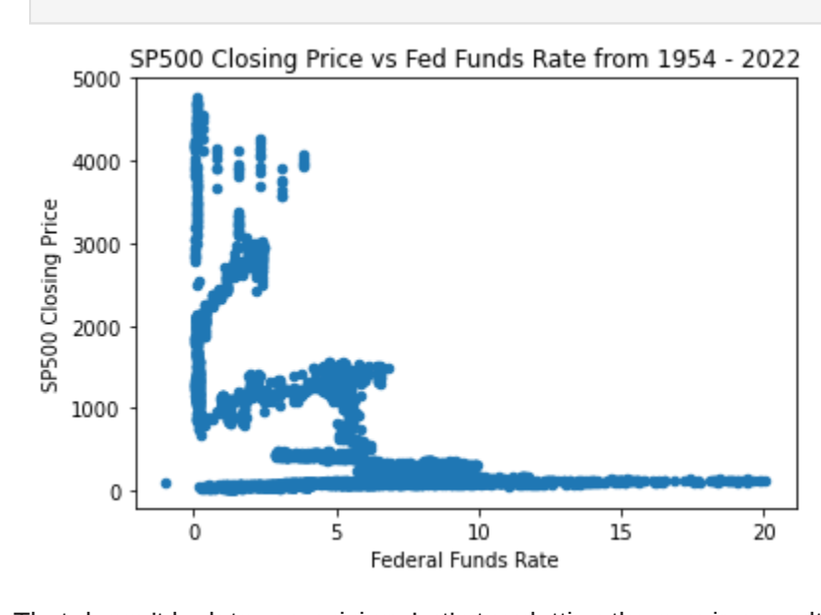
Now that the data has been processed, it's time to begin analysis.

3. DATA ANALYSIS

Regression

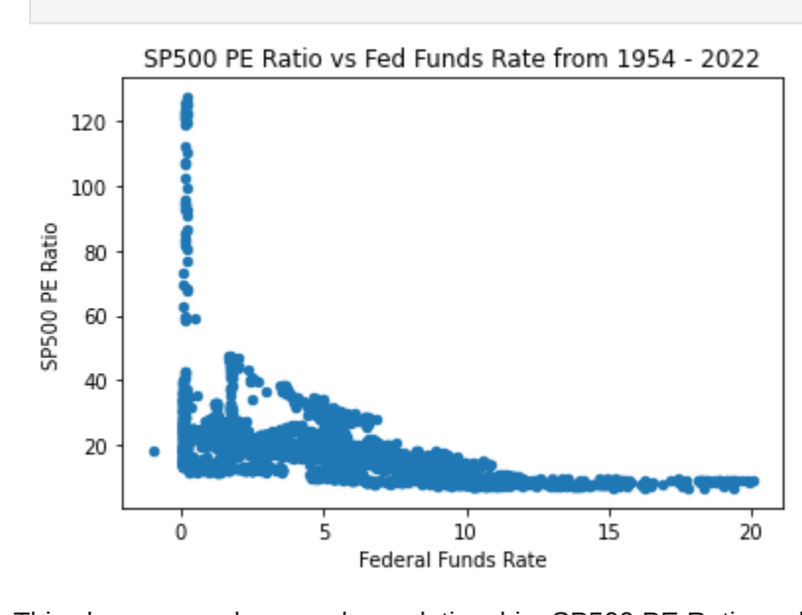
The relationship between interest rates and stock prices is further explored. It's unlikely the correlation will be very strong before making adjustments, since there are many factors that influence stock price beyond interest rates alone. But to get a basic understanding, it's not a bad idea to get a look at the relationship before any adjustments are made.

In [17]:
```
sp.plot.scatter(x = "Rate", y = "Close")
plt.title("SP500 Closing Price vs Fed Funds Rate from 1954 - 2022")
plt.xlabel("Federal Funds Rate")
plt.ylabel("SP500 Closing Price")
plt.show()
#A plot of SP500 closing price vs rates is made from the sp dataframe and labelled
```



That doesn't look too promising. Let's try plotting the earnings multiples, rather than price, instead.

In [18]:
```
sp.plot.scatter(x = "Rate", y = "PE Ratio")
plt.title("SP500 PE Ratio vs Fed Funds Rate from 1954 - 2022")
plt.xlabel("Federal Funds Rate")
plt.ylabel("SP500 PE Ratio")
plt.show()
#A plot of SP500 PE Ratio vs rates is made from the sp dataframe and labelled
```

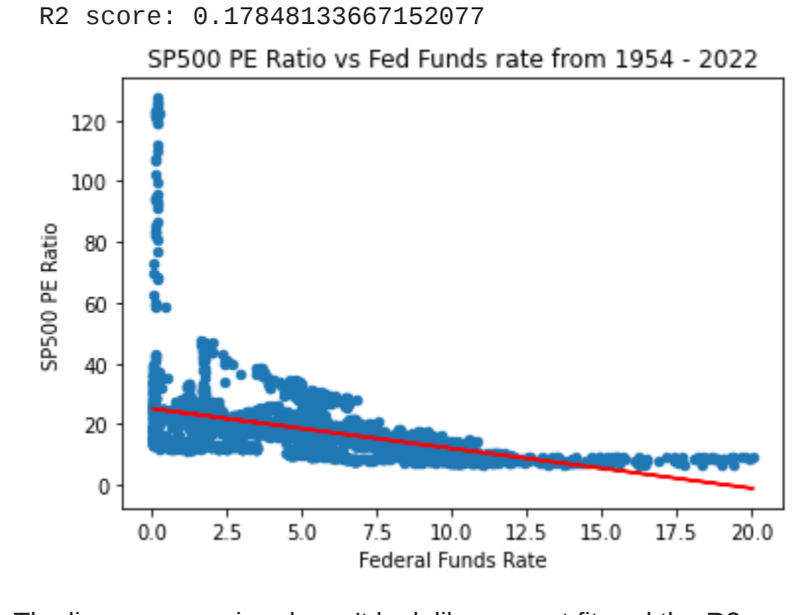


This shows a much more clear relationship. SP500 PE Ratio and federal funds rate appear to have an inverse correlation, and it looks to be exponential. Let's try a few different kinds of correlations and test which is best for modelling the data. It looks like there's a negative rate for one of the years, which is a clear outlier and likely a mistake since nominal rates have never been negative, this point will be dropped first.

In [19]:
```
sp = sp.drop(756)
sp = sp.reset_index()
del sp["index"]
#The datapoint with -1 rates is dropped and the dataframe is reindexed.
```

In [20]:
```
sp.plot.scatter(x = "Rate", y = "PE Ratio")
model = np.polyfit(sp["Rate"], sp["PE Ratio"], 1) #Create linear model from dataset
plt.plot(sp["Rate"], model[0]*sp["Rate"] + model[1], color = "red") #Plot model
predict = np.poly1d(model)
print("R2 score: " + str(r2_score(sp["PE Ratio"], predict(sp["Rate"])))) #Get r2 score of model and print it

plt.title("SP500 PE Ratio vs Fed Funds rate from 1954 - 2022")
plt.xlabel("Federal Funds Rate")
plt.ylabel("SP500 PE Ratio")
plt.show()
#A plot of SP500 PE Ratio vs rates is made from the sp dataframe and labelled
#The regression is plotted alongside the datapoints
```

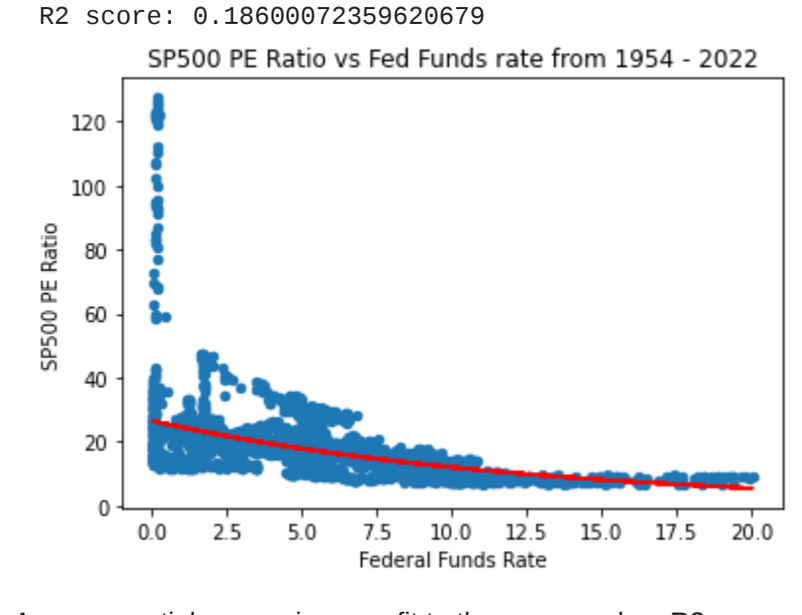
```
R2 score: 0.17848133667152077
```



The linear regression doesn't look like a great fit and the R2 score is quite low -- 0.178. Let's check an exponential regression next.

In [21]:
```
def monoExp(x, m, t, b):
    return m * np.exp(-t * x) + b
#This function just defines an exponential of the given parameters for fitting purposes

sp.plot.scatter(x = "Rate", y = "PE Ratio")

params, cv = curve_fit(monoExp, sp["Rate"], sp["PE Ratio"]) #Create exponential model of datapoints
m, t, b = params
plt.plot(sp["Rate"], monoExp(sp["Rate"], m, t, b), color = 'red') #Plot model

squaredDiffs = np.square(sp["PE Ratio"] - monoExp(sp["Rate"], m, t, b))
squaredDiffsFromMean = np.square(sp["PE Ratio"] - np.mean(sp["PE Ratio"]))
rSquared = 1 - np.sum(squaredDiffs) / np.sum(squaredDiffsFromMean)
print("R2 score: " + str(rSquared)) #Get r2 score of model and print it out

plt.title("SP500 PE Ratio vs Fed Funds rate from 1954 - 2022")
plt.xlabel("Federal Funds Rate")
plt.ylabel("SP500 PE Ratio")
plt.show()
#A plot of SP500 PE Ratio vs rates is made from the sp dataframe and labelled
#The regression is plotted alongside the datapoints
```

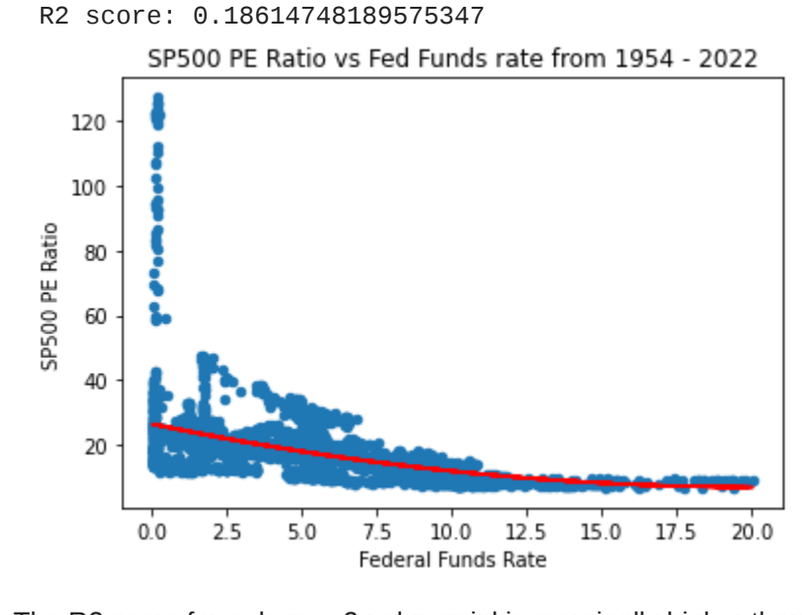
```
R2 score: 0.18600072359620679
```



An exponential regression was fit to the curve and an R2 score of 0.186 was generated. This is slightly higher than the linear regression R2 score, but still quite low. Perhaps we could do better with a polynomial regression?

In [22]:
```
sp.plot.scatter(x = "Rate", y = "PE Ratio")
model = np.polyfit(sp["Rate"], sp["PE Ratio"], 2) #Create quadratic model of datapoints
plt.plot(sp["Rate"], model[0]*sp["Rate"]**2 + model[1]*sp["Rate"] + model[2], color = "red") #Plot model
predict = np.poly1d(model)
print("R2 score: " + str(r2_score(sp["PE Ratio"], predict(sp["Rate"])))) #Find r2 score of model and print it out

plt.title("SP500 PE Ratio vs Fed Funds rate from 1954 - 2022")
plt.xlabel("Federal Funds Rate")
plt.ylabel("SP500 PE Ratio")
plt.show()
#A plot of SP500 PE Ratio vs rates is made from the sp dataframe and labelled
#The regression is plotted alongside the datapoints
```

```
R2 score: 0.18614748189575347
```



The R2 score for a degree 2 polynomial is marginally higher than an exponential fit, though they both round to 0.186.

Based on the low scores for all the types of models, although there seems to be an inverse correlation between interest rates and PE Ratio, it's difficult to pinpoint the exact relationship. It's more exponential/quadratic than it is linear, but only marginally so.

Also, since none of these models fit the data very well, it's fair to say there are confounding factors besides interest rates that also impact PE ratio. This makes sense, since a changing macroeconomic environment has resulted in vastly different investing conditions. For example, according to a paper from Columbia business school, globalization since the 1950s has resulted in steadily increasing PE ratios over time as foreign investors pile into US equity markets, driving valuations higher. Global financial markets are an extremely chaotic system, so while one factor (like rates) can certainly have an outsized influence on prices, it will never be possible to reduce the markets to a single variable.

Predictive Model

Can we use regression to predict the trajectory of PE ratios and stock prices given the rising interest rates of our day? Although the regressions did not have a strong correlation coefficient, there was still some correlation so it would potentially be valuable to at least explore this avenue.

We need some reference value to use for the Federal Funds Rate. The current interest rate is 3.78% according to https://fred.stlouisfed.org/series/FEDFUNDS. It is also expected to peak at 4.86% in March 2023, according to https://www.atlantafed.org/cenfis/market-probability-tracker, which uses futures markets to create this predicted rate. Let's take a look at what both of these rates imply.

The quadratic model will be used since it had the highest correlation coefficient.

In [23]:
```
rate_curr = 3.78
rate_mar = 4.86
#Defining relevant interest rates in each time period

PE_curr = model[0]*rate_curr**2 + model[1]*rate_curr + model[2]
PE_mar = model[0]*rate_mar**2 + model[1]*rate_mar + model[2]
rec_er = er.at[0, "S&P 500 EPS ($)"]
#Find the predicted current, march, and actual current earnings

curr_acc = str(sp.at[3569, "Close"])
curr_pred = str(PE_curr*float(rec_er))
curr_mar = str(PE_mar*float(rec_er))
#Use each earnings to find a price by mutliplying the pe ratio found by the price points at each time period
#Most recent earnings is used to find the current price

print("SP500 PE Ratio expected today: " + str(PE_curr))
print("SP500 PE Ratio expected in March 2023: " + str(PE_mar))
print()
print("Current SP500 Price: " + curr_acc)
print("SP500 Price expected today: " + curr_pred)
print("SP500 Price expected in March 2023: " + curr_mar)
```

```
SP500 PE Ratio expected today: 19.74832677508228
SP500 PE Ratio expected in March 2023: 18.102020320675962

Current SP500 Price: 3933.919921875
SP500 Price expected today: 3749.812288052623
SP500 Price expected in March 2023: 3437.2116184899514
```

According to this basic model, the SP500 should be trading about 200 points below the current price given the current interest rates, and is expected to drop below the current price by 500 points in the next three months. This should be taken with a grain of salt since the correlation coefficient was so low, but it will be interesting to see how accurate this prediction is.

Velocity of Policy

It's well-understood that markets are forward looking. So, change in conditions typically has greater influence on markets than the conditions themselves. For this reason, it would be interesting to explore a possible relationship between change in rates and stock valuations. This can also give some insight on how quickly policy impacts the financial markets.

To begin, it will be necessary to calculate weekly deltas for rates. This can be done by subtracting each week's rates from the last, in order to get a weekly rate of change in rates.

In [24]:
```
delta = []

for index, row in sp.iterrows():
    if(index > 0):
        delta.append(sp.at[index, "Rate"] - sp.at[index - 1, "Rate"])
#Iterate through the sp dataframe and find weekly delta in interest rates, append to an array

sp = sp.drop(0)
sp = sp.reset_index()
del sp["index"]
sp["Delta"] = delta
#Drop missing datapoint, reindex and create new column using array with delta of interest rates
```

In [25]:
```
sp
```

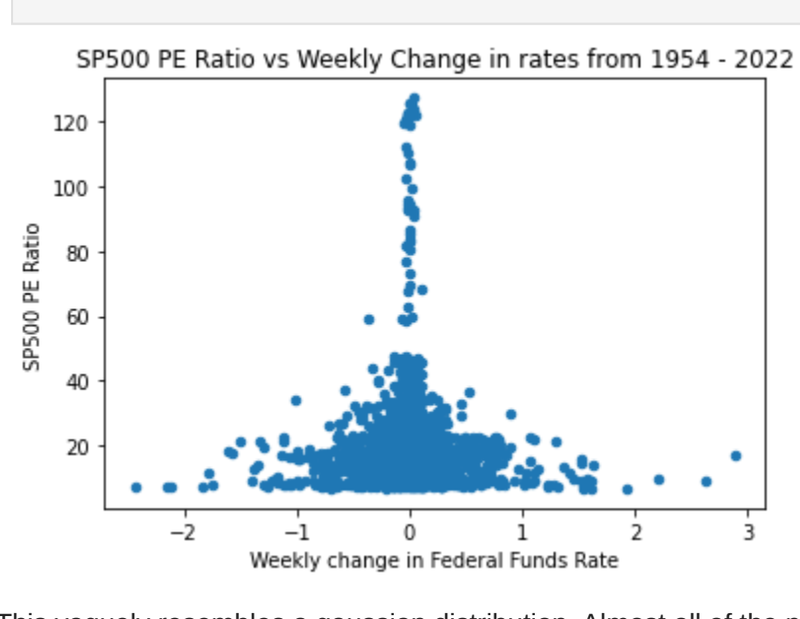
Out[25]:

	Date	Close	Rate	PE Ratio	Delta
0	07/12/54	30.059999	1.22	11.473282	0.22
1	07/19/54	30.309999	0.57	11.568702	-0.65
2	07/26/54	30.879999	0.63	11.786259	0.06
3	08/02/54	30.379999	0.27	11.551330	-0.36
4	08/09/54	30.719999	1.31	11.680608	1.04
...	...	...	...	...	...
3564	11/07/22	3992.929932	3.83	21.028702	0.75
3565	11/14/22	3965.340088	3.83	20.883401	0.00
3566	11/21/22	4026.120117	3.83	21.203498	0.00
3567	11/28/22	4071.699951	3.83	21.443543	0.00
3568	12/05/22	3933.919922	3.83	20.717927	0.00

3569 rows × 5 columns

The first week had to be dropped since it had no prior week to compare to in order to find a delta. Now, the PE ratio as a function of weekly delta can be plotted to get an idea of the trend we're working with.

In [26]:
```
sp.plot.scatter(x = "Delta", y = "PE Ratio")

plt.title("SP500 PE Ratio vs Weekly Change in rates from 1954 - 2022")
plt.xlabel("Weekly change in Federal Funds Rate")
plt.ylabel("SP500 PE Ratio")
plt.show()
#A plot of SP500 PE Ratio vs weekly change in rates is made from the sp dataframe and labelled
```

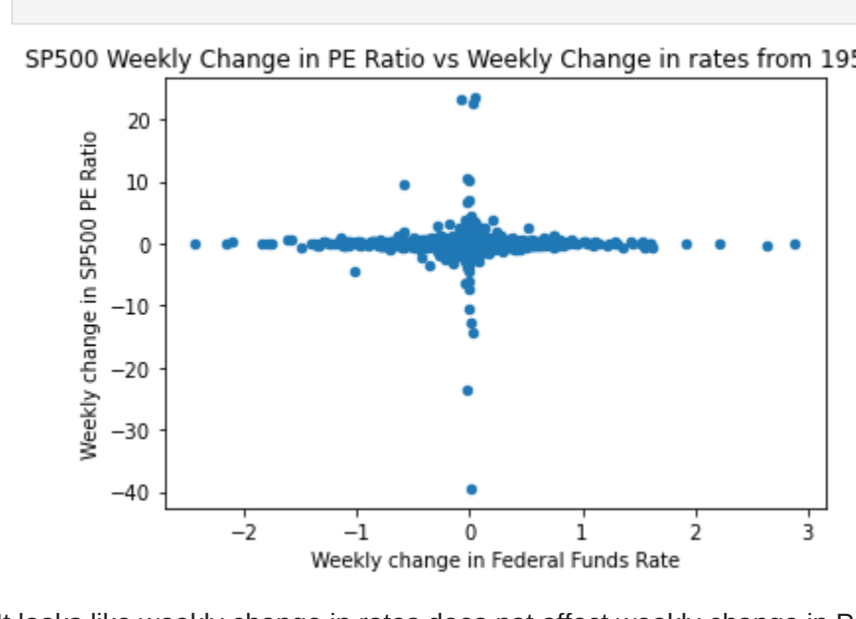


This vaguely resembles a gaussian distribution. Almost all of the points are near the origin, particularly all of the high PE Ratio datapoints. This makes sense since rates don't change for most weeks and indicates that a weekly change in rates of 0 is necessary for the periods of extremely high valuations in the stock market. The further from 0 in either direction, the lower the PE ratio. This is true when yields are rising since it means the cost of money is increasing which leads to lower valuations%20is,must%20outperform%20the%20hurdle%20rate.), and it's true when yields are falling since it means the Fed is trying to fight a recession which also brings with it compressed valuations.

This is fine, but it doesn't tell us how polciy actually impacts valuations, just what valuations are when policy decisions are made. For more insight, we need to explore how change in rates affects change in valuation.

In [27]:
```
delta_pe = []

for index, row in sp.iterrows():
    if(index > 0):
        delta_pe.append(sp.at[index, "PE Ratio"] - sp.at[index - 1, "PE Ratio"])
#sp dataframe is iterated through and weekly change in pe ratio is found and appended to delta_pe array

sp = sp.drop(0)
sp = sp.reset_index()
del sp["index"]
sp["Delta PE"] = delta_pe
#Drop missing datapoint, reindex and create new column using array with delta of pe ratios
```

In [28]:
```
sp.plot.scatter(x = "Delta", y = "Delta PE")

plt.title("SP500 Weekly Change in PE Ratio vs Weekly Change in rates from 1954 - 2022")
plt.xlabel("Weekly change in Federal Funds Rate")
plt.ylabel("Weekly change in SP500 PE Ratio")
plt.show()
#A plot of SP500 PE Ratio vs weekly change in weekly change in rates is made from the sp dataframe and labelled
```



It looks like weekly change in rates does not affect weekly change in PE ratios very much, since most of the points with a large weekly change in PE ratio occur when the change in rates is 0, and when interest rates are nonzero the weekly change in rates is minimal.

Weekly deltas in interest rates have no observable influence on stock valuations in the same timeframe, indicative of the slow rate with which monetary policy influences financial markets. This is well-understood since price typically lags behind policy.

4. HYPOTHESIS TESTING

A few hypothesis tests can be conducted, namely on the strength of correlation between interest rates and stock valuations. The null hypotheses is that interest rates and stock valuations are independent, and a low probability indicates that the alternative hypothesis is true (that there is a dependent relation).

A few tests can be utilized. The Chi-Squared test, Spearman's rank correlation, and Pearson's correlation coefficient are the ones which will be used in this case. The first tests whether two variables are related or independent, the second tests whether two samples have a monotonic relationship, while the third tests whether two samples have a linear relationship.

In [29]:
```
stat, p, dof, expected = chi2_contingency([sp["Rate"].tolist(), sp["PE Ratio"].tolist()])
stat2, p2 = spearmanr(sp["Rate"].tolist(), sp["PE Ratio"].tolist())
stat3, p3 = pearsonr(sp["Rate"].tolist(), sp["PE Ratio"].tolist())


print('ChiSquared stat=%.3f, p=%.3f' % (stat, p))
print('Spearman stat=%.3f, p=%.3f' % (stat2, p2))
print('Pearson stat=%.3f, p=%.3f' % (stat3, p3))
#Using scipy.stats functions, apply the statistical tests to the dataset and print out findings
```

```
ChiSquared stat=15151.813, p=0.000
Spearman stat=-0.570, p=0.000
Pearson stat=-0.423, p=0.000
```

In each test, a probability of 0.000 was found, indicating that the alternative hypothesis is likely to be true in each case. This means rates and PE ratios are related, and share a linear, monotonic relationship. This does appear to be true based on the nonzero correlation coefficient for a linear regression earlier, and also the shape of the graph which is monotonically decreasing.

There is clearly an inverse relationship between interest rates and stock valuations, but this relationship is not perfect due to the myriad of other factors that impact financial markets beyond interest rates.

5. CONCLUSION AND INSIGHTS

After analyzing stock valuations and interest rates, there is a clear inverse relationship between the two. The relationship is monotonic, with a stronger exponential and quadratic relationship than linear. All of the models had rather low correlation coefficients, indicative of confounding factors that also impact PE ratios beyond interest rates alone. Further, the weekly change in interest rates was correlated to PE ratios as well, in what appears to be a gaussian distribution. This is because the further from 0 change in rates in either direction, the lower the PE ratios tended to be. This was indicative of the fact that rates changing quickly are typically either a result of or in response to economic conditions which hamper financial markets. However valuations did not typically immediately respond to a weekly change in interest rates, indicative of a time delay between policy and price.

The relationship between interest rates and stock prices is established in the financial community, but these findings provide a bit more nuance on the nature of the relationship. It would be interesting to create a model that incorporates other variables beyond interest rates, such as macroeconomic or fiscal policy, in order to create a model with higher predictive power. Still, the strength of the correlation was remarkably robust to hypothesis testing, considering the simplicity of the model. It goes to show just how important the Fed Funds rate is to financial markets.