Does R&D Spending Improve Revenue for Companies?

In [1]:
```
#Packages
import pandas as pd
import numpy as np
import pandas as pd
import requests
import matplotlib.pyplot as plt
import time
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.preprocessing import RobustScaler, StandardScaler
import statsmodels.api as sm
from sklearn.cluster import KMeans
```

Data Preparation

Pull Data - API

In [2]:
```
# Header to pull API
headers = {'User-Agent': 'DSC680-project'}
```

In [3]:
```
# Needed data
metrics = {'Revenue': 'Revenues',
           'R&D': 'ResearchAndDevelopmentExpense',
           'Net_Income': 'NetIncomeLoss'}
```

In [4]:
```
# Range
years = range(2008, 2023)
```

In [5]:
```
# Empty dictionary
data = {}
```

In [6]:
```
# For loop to download each metric
for label, metric in metrics.items():
    # Empty list for each row
    rows = []


    # For loop to download each year for each company
    for year in years:
        # API - Change link for each matric and year
        url = f'https://data.sec.gov/api/xbrl/frames/us-gaap/{metric}/USD/CY{year}.json'
        # Request API
        response = requests.get(url, headers=headers)


        # Make sure request worked
        if response.status_code == 200:
            # Make webdata into dict
            data_pull = response.json()
            # Pull general account info for each company
            data_pull = data_pull['data']


            # For loop to store each company for each year
            for entry in data_pull:
                # Store only the wanted data
                rows.append({
                    # Company ID
                    'Company_ID': entry.get('cik'),
                    # Company Name
                    'Company_Name': entry.get('entityName'),
                    # Year
                    'Year': year,
                    # Metrics
                    label: entry.get('val'),
                    # Date it was filled
                    'Filing_Date': entry.get('end')})
        else:
            # Print Failer notice if it does not work
            print("Failed to retrieve")
    # Add
    time.sleep(0.15)

    # Make all metrics into data frames
    data[label] = pd.DataFrame(rows)
```

In [7]:
```
# Merge all three metrics into one dataframe
data_1 = data['Revenue'].merge(
    data['R&D'], on=['Company_ID', 'Company_Name', 'Year', 'Filing_Date'],
    how='outer').merge(data['Net_Income'], on=['Company_ID', 'Company_Name',
                                               'Year', 'Filing_Date'], how='outer')
```

In [8]:
```
data_1
```

Out[8]:

	Company_ID	Company_Name	Year	Revenue	Filing_Date	R&D	Net_Income
0	864328	BJ SERVICES CO	2008	5.359077e+09	2008-09-30	7.199700e+07	6.093650e+08
1	64978	MERCK SHARP & DOHME CORP.	2008	2.385030e+10	2008-12-31	4.805300e+09	7.808400e+09
2	1335793	CNX GAS CORP	2008	7.894210e+08	2008-12-31	NaN	2.390730e+08
3	868809	XTO ENERGY INC	2008	7.695000e+09	2008-12-31	NaN	1.912000e+09
4	1094316	TRINTECH GROUP PLC	2008	3.966400e+07	2009-01-31	6.069000e+06	-1.232000e+06
...	...	...	...	...	...	...	...
100402	2025410	StandardAero, Inc.	2022	NaN	2022-12-31	NaN	-2.100000e+07
100403	2037804	New Mountain Private Credit Fund	2022	NaN	2022-12-31	NaN	4.987100e+07
100404	2040127	KARMAN HOLDINGS INC.	2022	NaN	2022-12-31	NaN	-1.409862e+07
100405	2042694	Primo Brands Corp	2022	NaN	2022-12-31	NaN	-1.267000e+08
100406	2052959	Lionsgate Studios Corp.	2022	NaN	2023-03-31	NaN	-3.000000e+05

100407 rows × 7 columns

Clean Data

In [9]:
```
# Delete and duplicates
data_1 = data_1.drop_duplicates()
```

In [10]:
```
# Remove any Revenue that is 0 to predict sales growth.
# Do not know for sure that NaN means they did not spend money
data_2 = data_1.dropna(subset=['Revenue', 'Net_Income', 'R&D'])
```

Notes: Do not know for sure that NaN means no money was spent on R&D. After looking at SEC data it does not seemto be required. So NaN was dropped.

In [11]:
```
data_2
```

Out[11]:

	Company_ID	Company_Name	Year	Revenue	Filing_Date	R&D	Net_Income
0	864328	BJ SERVICES CO	2008	5.359077e+09	2008-09-30	7.199700e+07	6.093650e+08
1	64978	MERCK SHARP & DOHME CORP.	2008	2.385030e+10	2008-12-31	4.805300e+09	7.808400e+09
4	1094316	TRINTECH GROUP PLC	2008	3.966400e+07	2009-01-31	6.069000e+06	-1.232000e+06
10	890801	MCAFEE, INC.	2008	1.600065e+09	2008-12-31	2.520200e+08	1.722090e+08
12	758004	NOVELL INC	2008	9.565130e+08	2008-10-31	1.915470e+08	-8.745000e+06
...	...	...	...	...	...	...	...
45908	1990145	Holdco Nuvo Group D.G Ltd.	2022	0.000000e+00	2022-12-31	9.893000e+06	-2.067900e+07
45911	1991592	INLIF LIMITED	2022	6.652308e+06	2022-12-31	5.047110e+05	5.375550e+05
45913	1993727	SENSTAR TECHNOLOGIES CORPORATION	2022	3.555800e+07	2022-12-31	4.032000e+06	3.831000e+06
45915	1996862	BUNGE GLOBAL SA	2022	6.723200e+10	2022-12-31	3.300000e+07	1.610000e+09
45919	1999860	Wing Yip Food Holdings Group Limited	2022	1.307894e+08	2022-12-31	4.105172e+06	1.119398e+07

14024 rows × 7 columns

Notes: Large part of the data is NA. Also pull the industry because not all companies need R&D

In [12]:
```
# Make copy of data due to error
data_3 =  data_2.copy()
```

Unique CIKs - Pull data API

In [13]:
```
# Collect CIKs as a unique list
ciks = data_3['Company_ID'].astype(str).drop_duplicates().tolist()

# Empty list for each row
ciks_rows = []
```

In [14]:

```python
# Need CIKs code to be able to find industry
# For loop to pull each ID
for cik in ciks:
    # Add zeros before Company ID to follow SEC format
    cik_zero = cik.zfill(10)
    #link for each company ID
    url = f'https://data.sec.gov/submissions/CIK{cik_zero}.json'
    # Header
    headers = {'User-Agent': 'DSC680-project'}
    # Request data
    response = requests.get(url, headers=headers)


    # If everything is good, pull the wanted information
    if response.status_code == 200:
        # Make the files readable
        ciks_data = response.json()
        ciks_rows.append({
            # Company ID, need to merge
            'Company_ID': cik.lstrip('0'),
            # Classification code for SEC
            'SIC': ciks_data.get('sic'),
            # More Specific than Industry
            'SIC_Description': ciks_data.get('sicDescription'),
            # Pull Industry
            'Industry': ciks_data.get('ownerOrg'),
            # More company size (Not sure if needed)
            'Category': ciks_data.get('category')})
    # Skip company if it does not have it
    else:
        continue
    # Prevent error
    time.sleep(0.10)
```

In [15]:

```python
# Turn into DataFrame
ciks_data = pd.DataFrame(ciks_rows)
```

In [16]:

```python
ciks_data
```

Out[16]:

	Company_ID	SIC	SIC_Description	Industry	Category
0	864328	1389	Oil & Gas Field Services, NEC	None	Large Accelerated
1	64978	2834	Pharmaceutical Preparations	03 Life Sciences	
2	1094316	7372	Services-Prepackaged Software	None	
3	890801	7372	Services-Prepackaged Software	None	Large Accelerated Well Known Seasoned Issuer
4	758004	7372	Services-Prepackaged Software	None	Large Accelerated
...	...	...	...	...	...
2796	1983550	7389	Services-Business Services, NEC	07 Trade & Services	Non-accelerated filer Emerging growth company
2797	1984124	3317	Steel Pipe & Tubes	04 Manufacturing	Non-accelerated filer Emerging growth company
2798	1986247	2840	Soap, Detergents, Cleang Preparations, Perfume...	08 Industrial Applications and Services	Non-accelerated filer Emerging growth company
2799	1991592	3569	General Industrial Machinery & Equipment, NEC	06 Technology	Non-accelerated filer Emerging growth company
2800	1999860	2013	Sausages & Other Prepared Meat Products	04 Manufacturing	Non-accelerated filer Emerging growth company

2801 rows × 5 columns

In [17]:

```python
# Make sure both Company_ID is a str
data_3['Company_ID'] = data_3['Company_ID'].astype(str)
ciks_data['Company_ID'] = ciks_data['Company_ID'].astype(str)
```

In [18]:

```python
# Merge on Company_ID
data_4 = pd.merge(data_3, ciks_data, on='Company_ID', how='left')
```

In [19]:

```python
data_4
```

Out[19]:

	Company_ID	Company_Name	Year	Revenue	Filing_Date	R&D	Net_Income	SIC	SIC_Description	Industry	Category
0	864328	BJ SERVICES CO	2008	5.359077e+09	2008-09-30	7.199700e+07	6.093650e+08	1389	Oil & Gas Field Services, NEC	None	Large Accelerated
1	64978	MERCK SHARP & DOHME CORP.	2008	2.385030e+10	2008-12-31	4.805300e+09	7.808400e+09	2834	Pharmaceutical Preparations	03 Life Sciences	
2	1094316	TRINTECH GROUP PLC	2008	3.966400e+07	2009-01-31	6.069000e+06	-1.232000e+06	7372	Services-Prepackaged Software	None	
3	890801	MCAFEE, INC.	2008	1.600065e+09	2008-12-31	2.520200e+08	1.722090e+08	7372	Services-Prepackaged Software	None	Large Accelerated Well Known Seasoned Issuer
4	758004	NOVELL INC	2008	9.565130e+08	2008-10-31	1.915470e+08	-8.745000e+06	7372	Services-Prepackaged Software	None	Large Accelerated
...	...	...	...	...	...	...	...	...	...	...	...
14019	1990145	Holdco Nuvo Group D.G Ltd.	2022	0.000000e+00	2022-12-31	9.893000e+06	-2.067900e+07	3841	Surgical & Medical Instruments & Apparatus	08 Industrial Applications and Services	Non-accelerated filer Emerging growth company
14020	1991592	INLIF LIMITED	2022	6.652308e+06	2022-12-31	5.047110e+05	5.375550e+05	3569	General Industrial Machinery & Equipment, NEC	06 Technology	Non-accelerated filer Emerging growth company
14021	1993727	SENSTAR TECHNOLOGIES CORPORATION	2022	3.555800e+07	2022-12-31	4.032000e+06	3.831000e+06	3669	Communications Equipment, NEC	04 Manufacturing	Non-accelerated filer
14022	1996862	BUNGE GLOBAL SA	2022	6.723200e+10	2022-12-31	3.300000e+07	1.610000e+09	2070	Fats & Oils	04 Manufacturing	Large accelerated filer
14023	1999860	Wing Yip Food Holdings Group Limited	2022	1.307894e+08	2022-12-31	4.105172e+06	1.119398e+07	2013	Sausages & Other Prepared Meat Products	04 Manufacturing	Non-accelerated filer Emerging growth company

14024 rows × 11 columns

In [20]:
```
print(data_4.isna().sum())
```

```
Company_ID          0
Company_Name        0
Year                0
Revenue             0
Filing_Date         0
R&D                 0
Net_Income          0
SIC                 0
SIC_Description     0
Industry         4656
Category            0
dtype: int64
```

Notes: A lot of Industry are missing. Add Industry. No file found. Make a table. Links below to find information

Make Industry Dataset

In [21]:
```
# SIC Division
sic_divisions = [
    {"Division": "A", "Name": "Agriculture, Forestry, and Fishing", "Start": 100, "End": 999},
    {"Division": "B", "Name": "Mining", "Start": 1000, "End": 1499},
    {"Division": "C", "Name": "Construction", "Start": 1500, "End": 1799},
    {"Division": "D", "Name": "Manufacturing", "Start": 2000, "End": 3999},
    {"Division": "E", "Name": "Transportation, Communications, Electric, Gas & Sanitary", "Start": 4000, "End": 4999},
    {"Division": "F", "Name": "Wholesale Trade", "Start": 5000, "End": 5199},
    {"Division": "G", "Name": "Retail Trade", "Start": 5200, "End": 5999},
    {"Division": "H", "Name": "Finance, Insurance & Real Estate", "Start": 6000, "End": 6799},
    {"Division": "I", "Name": "Services", "Start": 7000, "End": 8999},
    {"Division": "J", "Name": "Public Administration", "Start": 9100, "End": 9729},
    {"Division": "-", "Name": "Nonclassifiable Establishments", "Start": 9900, "End": 9999}]
```

Links:
https://www.naics.com/sic-codes-industry-drilldown/,
https://siccode.com/sic-code-lookup-directory,
https://fieldtexcases.com/blog/manufacturing-sic-codes/?utm_source=chatgpt.com

In [22]:
```
# Make a funcation to make a table of all codes
def get_division(sic):
    for d in sic_divisions:
        #only make a list between the SIC cat. codes
        if d["Start"] <= sic <= d["End"]:
            # Return results
            return d["Division"], d["Name"]
    # Give no result if it does not meet the requirements
    return None, None
```

In [23]:
```
# Funcation to make the numbers from get_division into int format
def int_division(x):
    try:
        # Make results an int.
        x_int = int(x)
        # Return results as a pd series for dataframe
        return pd.Series(get_division(x_int))
    # If error occurs enter none
    except Exception:
        return pd.Series([None, None])
```

In [24]:
```
data_4[["SIC_Division", "SIC_Industry"]] = data_4["SIC"].apply(int_division)
```

In [25]:
```
data_4.head()
```

Out[25]:

	Company_ID	Company_Name	Year	Revenue	Filing_Date	R&D	Net_Income	SIC	SIC_Description	Industry	Category	SIC_Division	SI
0	864328	BJ SERVICES CO	2008	5.359077e+09	2008-09-30	7.199700e+07	6.093650e+08	1389	Oil & Gas Field Services, NEC	None	Large Accelerated	B	
1	64978	MERCK SHARP & DOHME CORP.	2008	2.385030e+10	2008-12-31	4.805300e+09	7.808400e+09	2834	Pharmaceutical Preparations	03 Life Sciences		D	Ma
2	1094316	TRINTECH GROUP PLC	2008	3.966400e+07	2009-01-31	6.069000e+06	-1.232000e+06	7372	Services-Prepackaged Software	None		I	
3	890801	MCAFEE, INC.	2008	1.600065e+09	2008-12-31	2.520200e+08	1.722090e+08	7372	Services-Prepackaged Software	None	Large Accelerated Well Known Seasoned Issuer	I	
4	758004	NOVELL INC	2008	9.565130e+08	2008-10-31	1.915470e+08	-8.745000e+06	7372	Services-Prepackaged Software	None	Large Accelerated	I	

Clean Data

In [26]:
```
# Calculate the sales growth
# sort to make the results sort each year for each company
data_5 = data_4.sort_values(['Company_ID', 'Year'])
# Make new column for sales growth for revenue
data_5['Sales_Growth'] = data_4.groupby('Company_ID')['Revenue'].pct_change()
# Make new column for sales growth for R&D
data_5['RD_Growth'] = data_4.groupby('Company_ID')['R&D'].pct_change()
```

In [27]:
```
# Check for NaN
print(data_5.isna().sum())
```

```
Company_ID            0
Company_Name          0
Year                  0
Revenue               0
Filing_Date           0
R&D                   0
Net_Income            0
SIC                   0
SIC_Description       0
Industry           4656
Category              0
SIC_Division         11
SIC_Industry         11
Sales_Growth       3187
RD_Growth          2885
dtype: int64
```

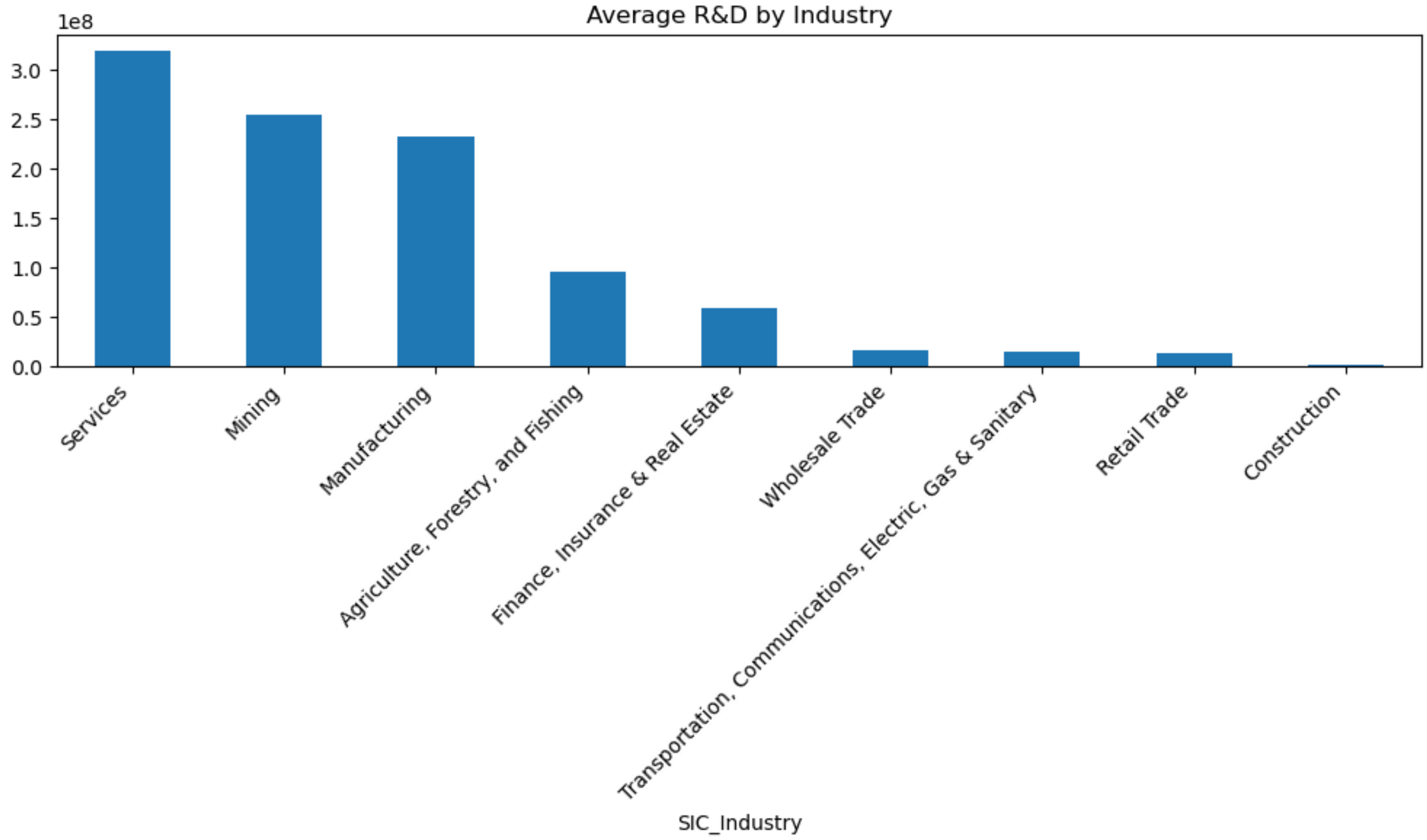
In [28]:
```
# Drop NaN
data_5 = data_5.dropna(subset=['Industry','SIC_Division', 'SIC_Industry', 'Sales_Growth'])
```

Do companies that currently spend heavily on R&D achieve faster sales growth than companies spending less?

In [29]:
```
# Group by industry and calculate average sales growth
industry_growth = data_5.groupby('SIC_Industry')['R&D'].mean().sort_values(ascending=False)
```

Average Sales Growth by Industry

In [99]:
```
# Graph size
plt.figure(figsize=(10, 6))
# Make a bar chart
industry_growth.plot(kind='bar')
plt.title('Average R&D by Industry')
# Make industry easy to read
plt.xticks(rotation=45, ha='right')
# Make labels print within size
plt.tight_layout()
# Print graph
plt.show()
```



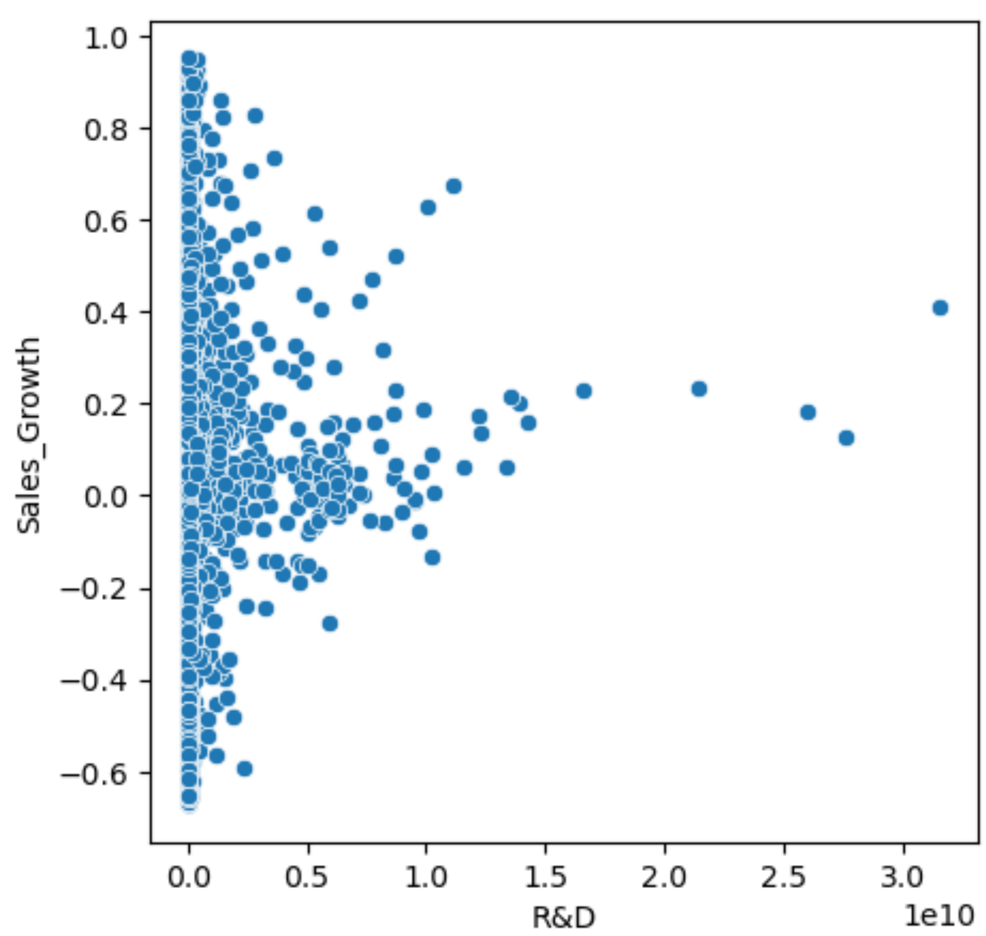
In [31]:
```
# Fix error. Remove outliers
# 25% of the data Q1
Q1 = data_5['Sales_Growth'].quantile(0.25)
# 75% of the data Q3
Q3 = data_5['Sales_Growth'].quantile(0.75)
# Find the spread (IQR)
spread = Q3 - Q1
```

In [32]:
```
# find lower outliers
lower_bound = Q1 - 1.5 * spread
# Find upper outliers
upper_bound = Q3 + 1.5 * spread
```

In [33]:
```
# Filter out outliers
data_6 = data_5[(data_5['Sales_Growth'] >= lower_bound) & (data_5['Sales_Growth'] <= upper_bound)]
```

R&D Spending Vs Sales Growth - Scatterplot

In [34]:
```
# Figure Size
plt.figure(figsize=(5,5))
# Make a scatter plot
sns.scatterplot(x='R&D', y='Sales_Growth', data=data_6)
# Print graph
plt.show()
```



Notes: Try Random forest. results are not linear

Random Forest Regressor

In [35]:
```
# Make copy to prevent error
data_rnd = data_6.copy()
```

In [36]:
```
# Define features and targets to predict if current sales growth
X = data_rnd[['R&D']]
y = data_rnd['Sales_Growth']
```

In [37]:
```
# Train Model
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [38]:
```
# Allow random forest
rf_model = RandomForestRegressor(random_state=42)
```

In [39]:
```
# Fit model
rf_model.fit(X_train, y_train)
```

Out[39]:
```
RandomForestRegressor
RandomForestRegressor(random_state=42)
```

In [40]:
```
# Predict model
y_pred_rf = rf_model.predict(X_test)
```

In [41]:
```
print("Random Forest - Just R&D")
print("MAE:", mean_absolute_error(y_test, y_pred_rf))
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred_rf)))
print("R2:", r2_score(y_test, y_pred_rf))
```

```
Random Forest - Just R&D
MAE: 0.26508584969110555
RMSE: 0.348804499710147
R2: -0.44151184759756235
```

In [42]:
```
# Define features and targets to predict if current sales growth
# include year to account for covid and industry due
X = pd.get_dummies(data_rnd[['R&D', 'Year', 'SIC_Industry']])
y = data_rnd['Sales_Growth']
```

In [43]:
```
# Train Model
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [44]:
```
# Allow random forest
rf_model = RandomForestRegressor(random_state=42)
```

In [45]:
```
# Fit model
rf_model.fit(X_train, y_train)
```

Out[45]:
```
RandomForestRegressor
RandomForestRegressor(random_state=42)
```

In [46]:
```
# Predict model
y_pred_rf = rf_model.predict(X_test)
```

In [47]:
```
print("Random Forest - R&D, Year, SIC_Industry")
print("MAE:", mean_absolute_error(y_test, y_pred_rf))
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred_rf)))
print("R2:", r2_score(y_test, y_pred_rf))
```

```
Random Forest - R&D, Year, SIC_Industry
MAE: 0.24409953992254385
RMSE: 0.3254016994114377
R2: -0.25456651508392625
```

Not a good model. Try robust scaler for R&D and sales growth to decrease skew

In [48]:
```
# Copy of data for scaled data
data_7 = data_6.copy()
```

In [49]:
```
# Allow scaler
scaler = RobustScaler()
```

In [50]:
```
# Scale R&D and sales growth
data_7['R&D_scaled'] = scaler.fit_transform(data_7[['R&D']])
data_7['Sales_Growth_scaled'] = scaler.fit_transform(data_7[['Sales_Growth']])
```

In [51]:
```
# Define features and targets to predict
X = pd.get_dummies(data_7[['R&D_scaled', 'Year', 'SIC_Industry']])
y = data_7['Sales_Growth_scaled']
```

In [52]:
```
# Train Model
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [53]:
```
# Fit model
rf_model.fit(X_train, y_train)
```

Out[53]:
```
RandomForestRegressor
RandomForestRegressor(random_state=42)
```

In [54]:
```
# Predict model
y_pred_rf = rf_model.predict(X_test)
```

In [55]:
```
print("Random Forest")
print("MAE:", mean_absolute_error(y_test, y_pred_rf))
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred_rf)))
print("R2:", r2_score(y_test, y_pred_rf))
```

```
Random Forest
MAE: 0.8770134832587054
RMSE: 1.1695590078350582
R2: -0.2560033951876832
```

Model is even worse. Only look at the top three

In [56]:
```
# Top three sales growth
keep = ['Agriculture, Forestry, and Fishing','Manufacturing','Services']
```

In [57]:
```
# New data set of only top three
data_8 = data_7[data_7['SIC_Industry'].isin(keep)]
```

In [58]:
```
# Make copy to prevent error
data_rnd_top3 = data_8.copy()
```

In [59]:
```
# Define features and targets to predict
X = data_rnd_top3[['R&D']]
y = data_rnd_top3['Sales_Growth']
```

In [60]:
```
# Train Model
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [61]:
```
# Fit model
rf_model.fit(X_train, y_train)
```

Out[61]:
```
RandomForestRegressor
RandomForestRegressor(random_state=42)
```

In [62]:
```
# Predict model
y_pred_rf = rf_model.predict(X_test)
```

In [63]:
```
# Notes: Top three
print("Random Forest")
print("MAE:", mean_absolute_error(y_test, y_pred_rf))
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred_rf)))
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred_rf)))
print("R2:", r2_score(y_test, y_pred_rf))
```

```
Random Forest
MAE: 0.267700751227656
RMSE: 0.35298014074797307
RMSE: 0.35298014074797307
R2: -0.5604123043938596
```

Try a different model

Random Forest - Only looking at R&D as Indep. MAE: 0.21793900803113708 RMSE: 0.28644902589828736 RMSE: 0.28644902589828736 R2: -0.23215625461103118

Random Forest - Three indep. varb MAE: 0.2049755338401916 RMSE: 0.27509560160027885 R2: -0.1364187217596715

Gradient Boosting

In [64]:
```
# Define features and targets to predict
X = data_7[['R&D']]
y = data_7['Sales_Growth']
```

In [65]:
```
# Allow Gradient Boosting
gb_model = GradientBoostingRegressor(random_state=42)
```

In [66]:
```
# Fit the model
gb_model.fit(X_train, y_train)
```

Out[66]:
```
GradientBoostingRegressor
GradientBoostingRegressor(random_state=42)
```

In [67]:
```
# Predict
y_pred_gb = gb_model.predict(X_test)
```

In [68]:
```
print("Gradient Boosting - R&D")
print("MAE:", mean_absolute_error(y_test, y_pred_gb))
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred_gb)))
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred_gb)))
print("R2:", r2_score(y_test, y_pred_gb))
```

```
Gradient Boosting - R&D
MAE: 0.20834690476344978
RMSE: 0.28570668025628765
RMSE: 0.28570668025628765
R2: -0.022302945878913905
```

In [69]:
```
# Define features and targets to predict
X = pd.get_dummies(data_rnd[['R&D', 'Year', 'SIC_Industry']])
y = data_rnd['Sales_Growth']
```

In [70]:
```
# Train Model
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [71]:
```
# Fit the model
gb_model.fit(X_train, y_train)
```

Out[71]:
```
GradientBoostingRegressor
GradientBoostingRegressor(random_state=42)
```

In [72]:
```
# Predict
y_pred_gb = gb_model.predict(X_test)
```

In [73]:
```
print("Gradient Boosting - Top Three with Sales Growth")
print("MAE:", mean_absolute_error(y_test, y_pred_gb))
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred_gb)))
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred_gb)))
print("R2:", r2_score(y_test, y_pred_gb))
```

```
Gradient Boosting - Top Three with Sales Growth
MAE: 0.20850410790856033
RMSE: 0.289767258403366
RMSE: 0.289767258403366
R2: 0.005161275544728672
```

Notes: Model is better. Look at Inustry and Year

In [74]:
```
# Define features and targets to predict. Look at all
X = pd.get_dummies(data_7[['R&D', 'Year', 'SIC_Industry']])
y = data_7['Sales_Growth']
```

In [75]:
```
# Train Model
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [76]:
```
# Fit model
gb_model.fit(X_train, y_train)
```

Out[76]:
```
GradientBoostingRegressor
GradientBoostingRegressor(random_state=42)
```

In [77]:
```
# Predict
y_pred_gb = gb_model.predict(X_test)
```

In [78]:
```
print("Gradient Boosting - All Industry")
print("MAE:", mean_absolute_error(y_test, y_pred_gb))
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred_gb)))
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred_gb)))
print("R2:", r2_score(y_test, y_pred_gb))
```

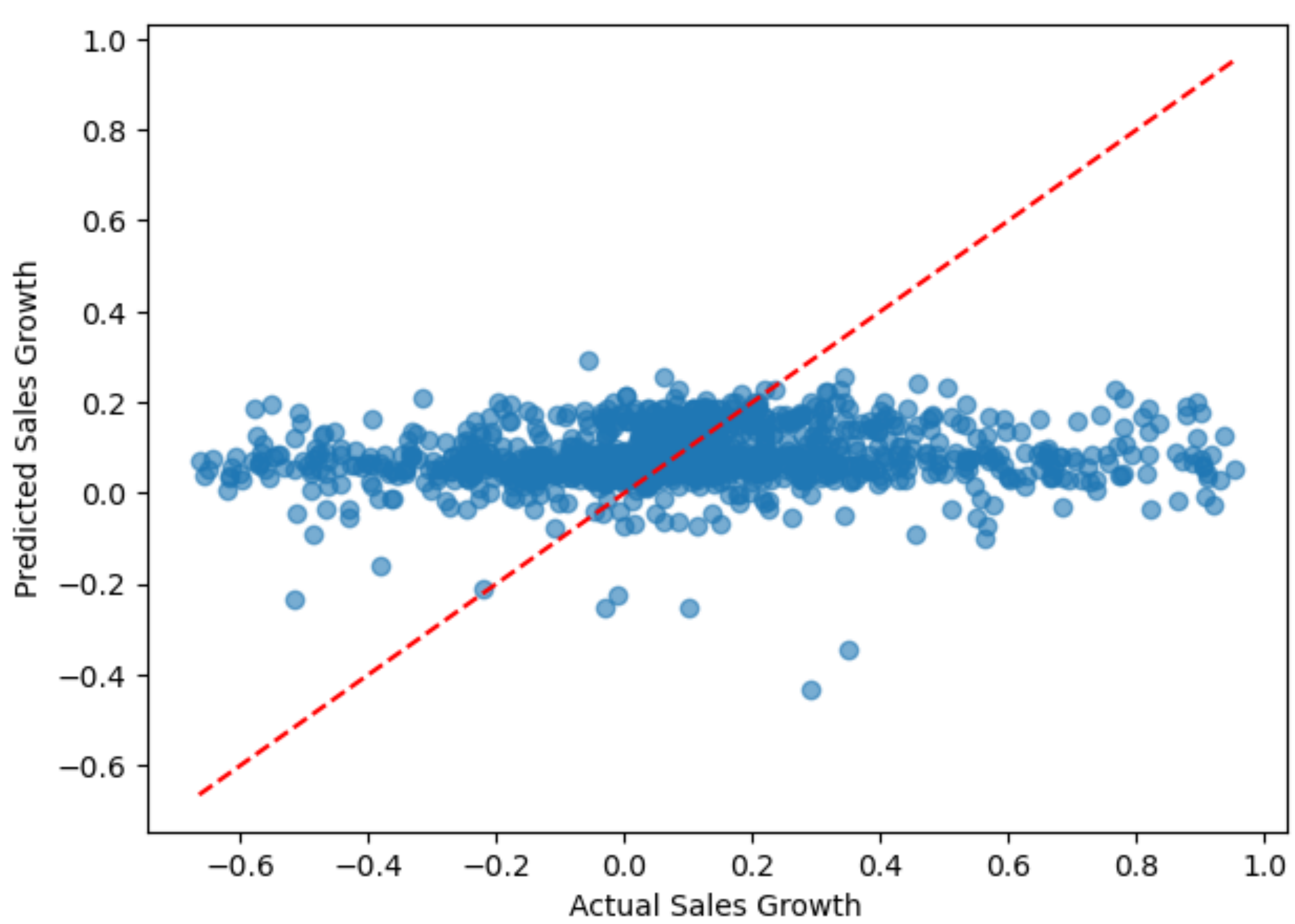
```
Gradient Boosting - All Industry
MAE: 0.20850410790856033
RMSE: 0.289767258403366
RMSE: 0.289767258403366
R2: 0.005161275544728672
```

Model is the same. Does not look like it made any difference to look at the top three or all of them.

R&D, year, and industry only explained a small amount of variation in sales growth

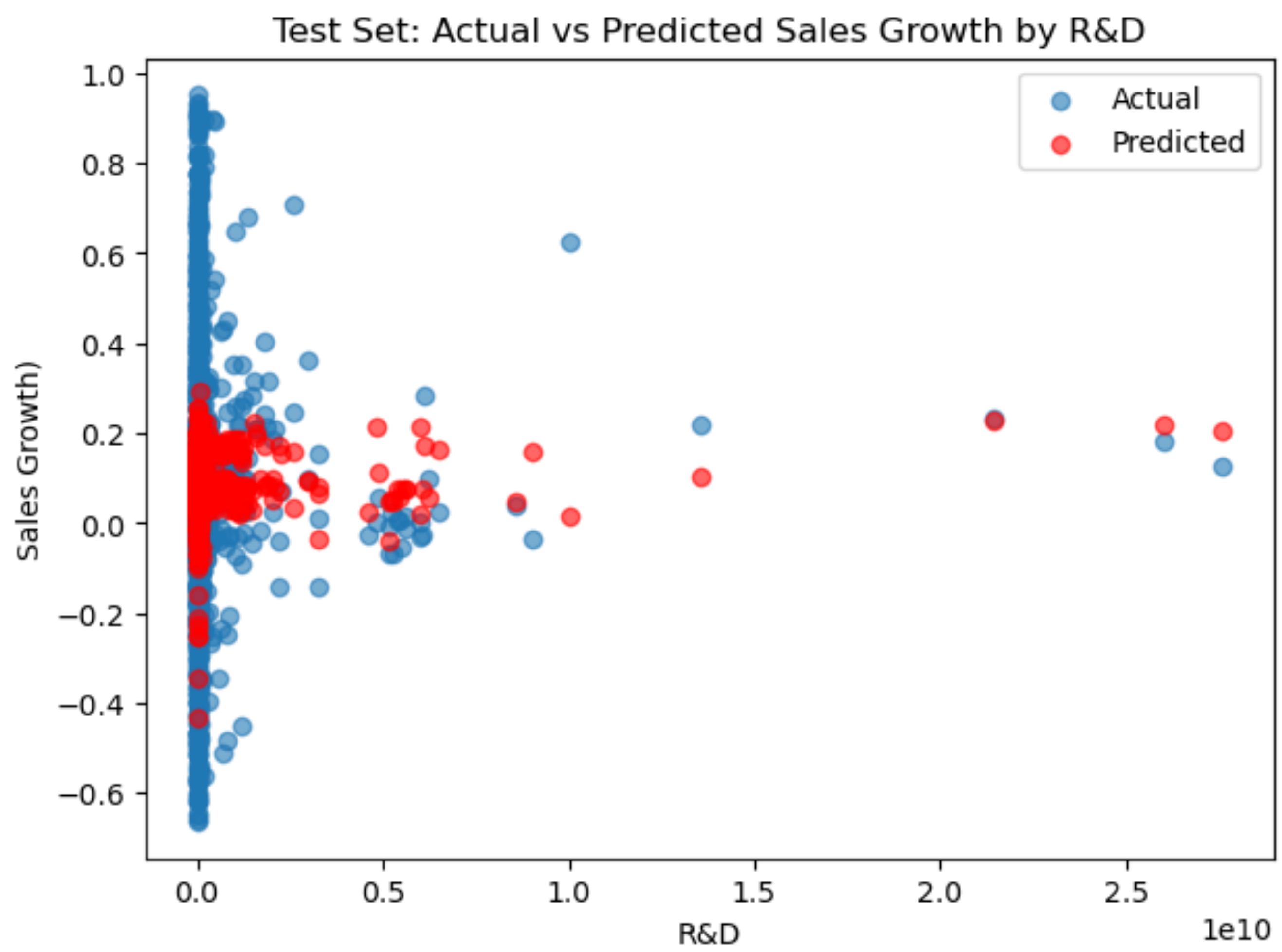
Actual Sales Growth Vs. Predicted

In [79]:
```
# Figure size
plt.figure(figsize=(7, 5))
# Make scatter plot
plt.scatter(y_test, y_pred_gb, alpha=0.6)
# X-label
plt.xlabel('Actual Sales Growth')
# Y-label
plt.ylabel('Predicted Sales Growth')
# Plot graph
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], '--', color='red')
# Print graph
plt.show()
```



Test Set Actual vs Predicted Sales Growth - Scatterplot

In [80]:
```
# Figure size
plt.figure(figsize=(7, 5))
# Scatterplot of Actual Data
plt.scatter(X_test['R&D'], y_test, label='Actual', alpha=0.6)
#Scatterplot of Predicted data
plt.scatter(X_test['R&D'], y_pred_gb, label='Predicted', alpha=0.6, color='red')
# X- Label
plt.xlabel('R&D')
#Y-Label
plt.ylabel('Sales Growth)')
# Titel
plt.title('Test Set: Actual vs Predicted Sales Growth by R&D')
#Key
plt.legend()
#Print Graph
plt.show()
```



When a company increases its R&D spending in one year, does its sales growth accelerate in the subsequent year?

Notes: Not enough data per company for AIRMA. Use OLS

In [81]:
```
#Make copy of data
data_9 = data_8.copy()
```

In [82]:
```
# Group by Company ID and R&D. Shift one year
data_9['R&D'] = data_9.groupby('Company_ID')['R&D'].shift(1)
```

In [83]:
```
# Drop Na
data_9 = data_9[['Revenue', 'R&D', 'SIC_Industry']].dropna()
```

In [84]:
```
# Revenue ~ Prior year R&D
# Independ. Varb.
X = data_9[['R&D']]
# Add intercept
X = sm.add_constant(X)
# Depend. Varb.
y = data_9['Revenue']
```

In [85]:
```
# Fit Model
model = sm.OLS(y, X).fit()
```

In [86]:
```
print(model.summary())
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                Revenue   R-squared:                       0.243
Model:                            OLS   Adj. R-squared:                  0.243
Method:                 Least Squares   F-statistic:                     1388.
Date:                Sun, 13 Jul 2025   Prob (F-statistic):          1.04e-263
Time:                        17:14:25   Log-Likelihood:            -1.0940e+05
No. Observations:                4331   AIC:                         2.188e+05
Df Residuals:                    4329   BIC:                         2.188e+05
Df Model:                           1                                         
Covariance Type:            nonrobust                                         
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       3.241e+09   3.53e+08      9.175      0.000    2.55e+09    3.93e+09
R&D           10.1508      0.272     37.251      0.000       9.617      10.685
==============================================================================
Omnibus:                     7217.325   Durbin-Watson:                   0.374
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          5339614.245
Skew:                          11.224   Prob(JB):                         0.00
Kurtosis:                     173.544   Cond. No.                     1.33e+09
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.33e+09. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Notes:

- R&D coefficient = 10.15 | significant because p value is also $p < .001$
- R^2= 0.274 | means 27.4% of revenue variation explained by prior year's R&D
- Each 1 million increase in prior year's R&D spending is associated with an average increase of about 10.15 million in revenue the following year.
- The relationship is statistically significant ($p < 0.001$), suggesting a real effect.
- $R^2 = 0.27$: Prior year R&D explains a portion of future revenue, but there are still many other factors.

Classify companies into distinct groups, e.g., high investors, moderate investors, and low investors, based on their R&D spending and sales growth patterns.

In [87]:
```
# Group companies by mean R&D and Sales Growth
data_k = data_8.groupby('Company_ID').agg({'R&D': 'mean',
                                          'Sales_Growth': 'mean'}).dropna()
```

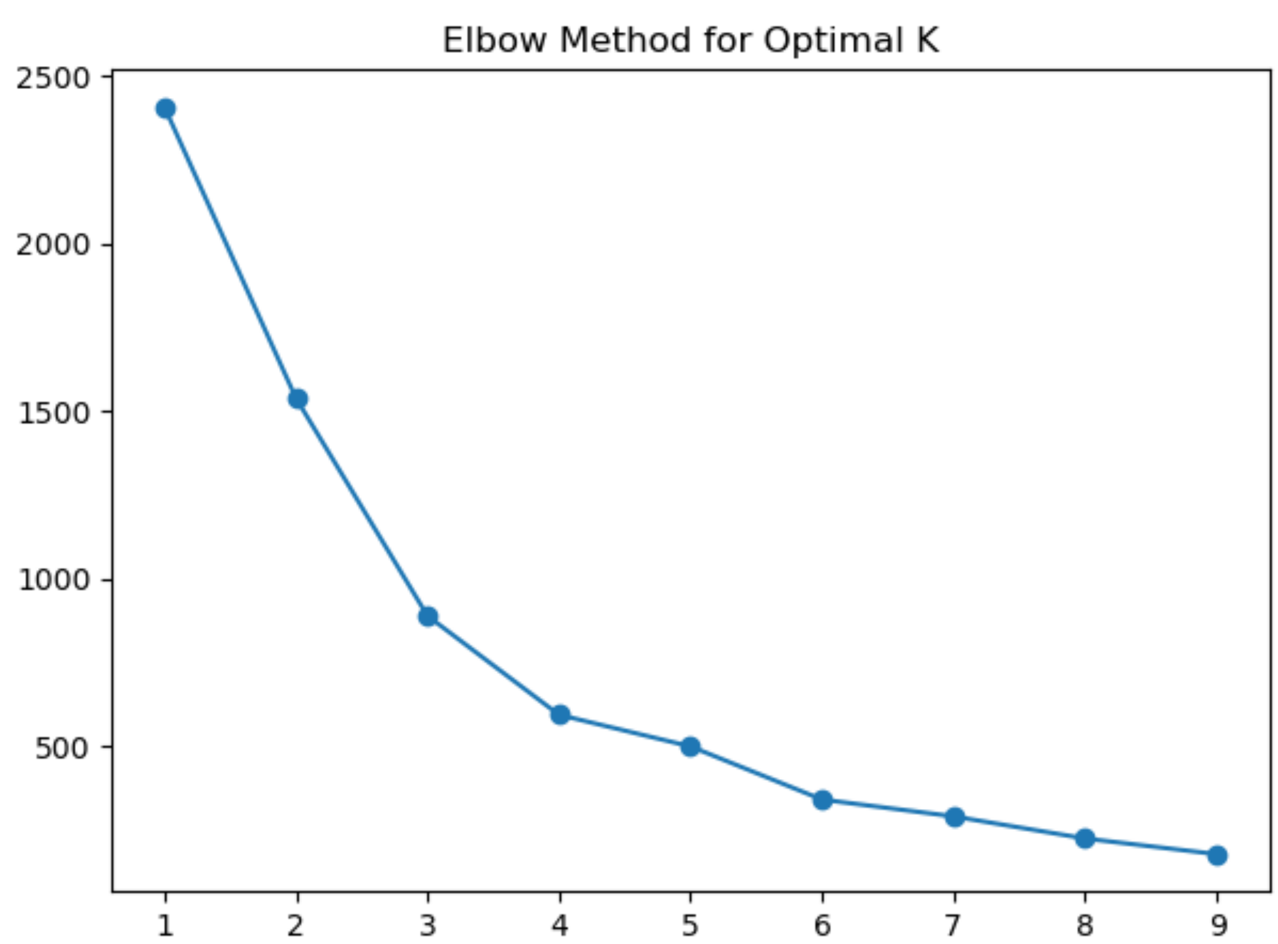
In [88]:
```
# Scale, fit and transform the
X_optK = StandardScaler().fit_transform(data_k[['R&D', 'Sales_Growth']])
```

In [89]:
```
# Empty list for k clusters
k_clus = []
```

Elbow Graph - Find K-Cluster

In [90]:
```
# Find Optimal Clster
for k in range(1, 10):
    # For each cluster fit a Kmeans model
    km = KMeans(n_clusters=k, random_state=42).fit(X_optK)
    # Append list
    k_clus.append(km.inertia_)
```

In [91]:
```
# Figure size
plt.figure(figsize=(7, 5))
# Plot the k clusters and mark each one
plt.plot(range(1,10), k_clus, marker='o')
# Title
plt.title('Elbow Method for Optimal K')
# Print Graph
plt.show()
```



In [92]:
```
# Allow KMeans clustering
kmeans = KMeans(n_clusters=4, random_state=42)
```

In [93]:
```
# Data clusters to data table
data_k['Cluster'] = kmeans.fit_predict(data_k[['R&D', 'Sales_Growth']])
```

In [94]:
```
data_k.head(5)
```

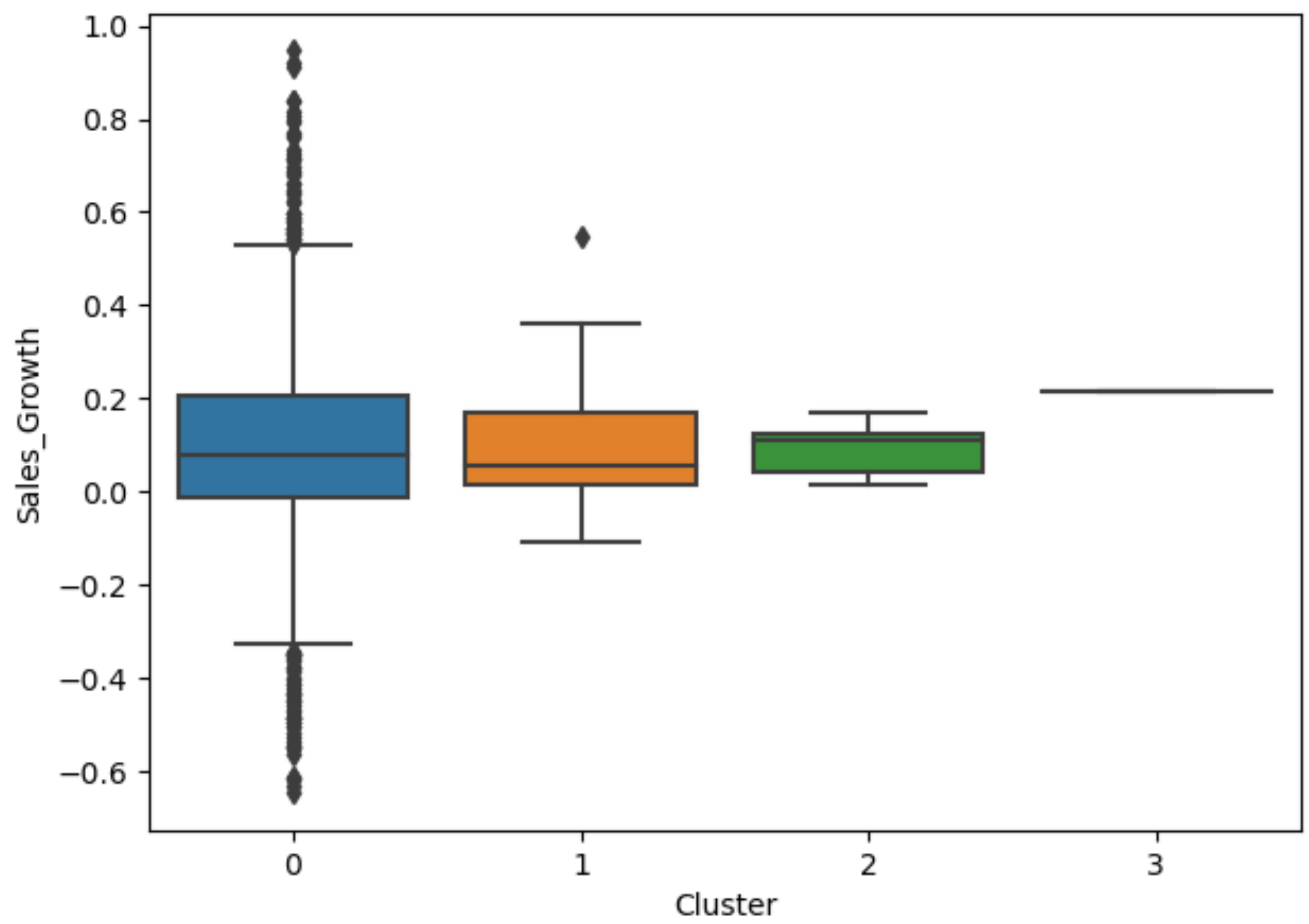
Out[94]:

Company_ID	R&D	Sales_Growth	Cluster
1000694	2.629199e+08	0.111902	0
1001115	1.495078e+07	-0.015650	0
1001233	5.111557e+07	0.098167	0
1001907	3.301600e+06	0.049600	0
1002047	6.708333e+08	0.224641	0

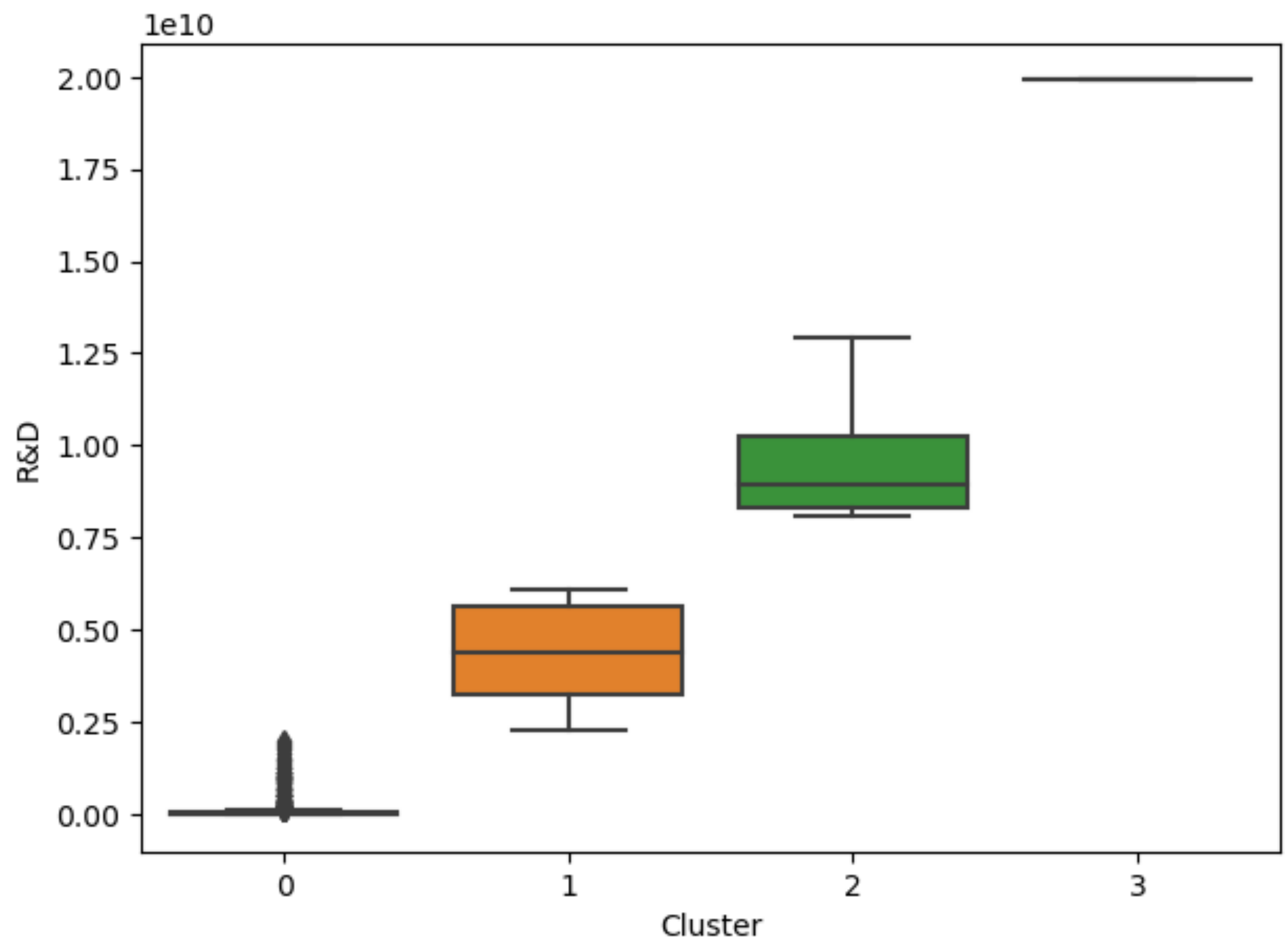
In [95]:
```
data_k['Cluster'].unique()
```

Out[95]:
```
array([0, 1, 2, 3], dtype=int32)
```

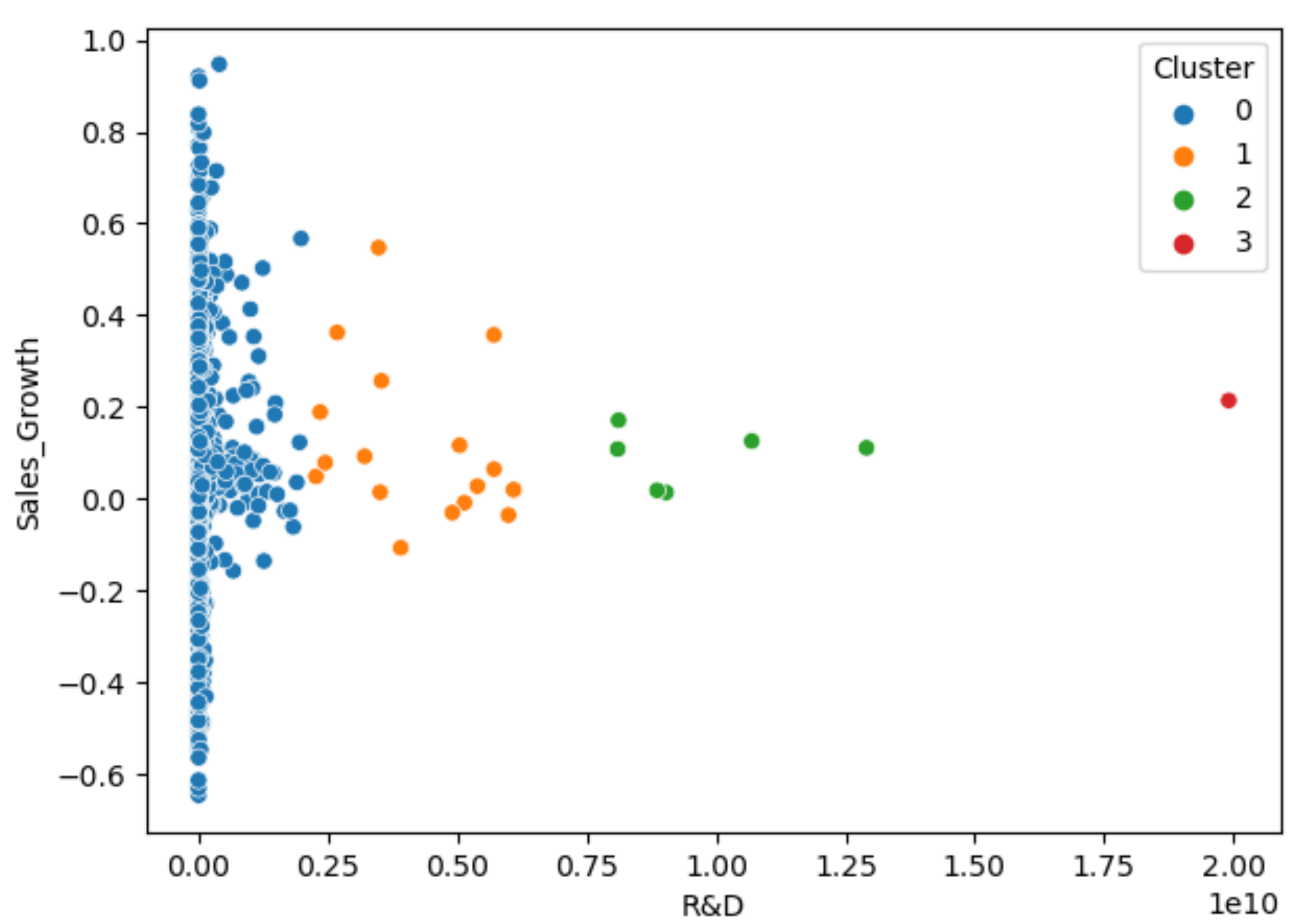
In [96]:
```
# Figure size
plt.figure(figsize=(7, 5))
# Make a box blot of the clusters (Sales Growth)
sns.boxplot(x='Cluster', y='Sales_Growth', data=data_k)
# Print graph
plt.show()
```



In [97]:
```
# Figure size
plt.figure(figsize=(7, 5))
# Make a box blot of the clusters (R&D)
sns.boxplot(x='Cluster', y='R&D', data=data_k)
# Print Graph
plt.show()
```



In [98]:
```
# Figure size
plt.figure(figsize=(7, 5))
# Scatter Plot of all the clusters
sns.scatterplot(x='R&D', y='Sales_Growth', hue='Cluster', data=data_k, palette='tab10')
# Print Graph
plt.show()
```



In [ ]: