```python
import pandas as pd
import numpy as np

# Load the dataset
data_path = 'Final Data CA+FA.csv'
data = pd.read_csv(data_path)

# Display the first few rows of the dataframe to understand its
structure
data.head()
```

```
                WOS ID First Name                   Country Country
Code  \
0  WOS:000174718100007     Janice                   England
GB
1  WOS:000207062600010   Kyoungho               South Korea
KR
2  WOS:000207451700010  Elizabeth  United States of America
US
3  WOS:000207695900002    Joachim  United States of America
US
4  WOS:000207784200003      Beate                   Germany
DE

   Gender Gender Probability  Publication Year    Author Type
0  female            100.00%              2002  Corresponding
1    male            100.00%              2007  Corresponding
2  female            100.00%              2007  Corresponding
3    male            100.00%              2008  Corresponding
4  female             99.00%              2009  Corresponding
```

```python
# Count unique WOS ID occurrences per year
publication_counts = data.groupby('Publication Year')['WOS
ID'].nunique()

# Convert the series to a DataFrame
publication_counts_df = publication_counts.reset_index()
publication_counts_df.columns = ['Year', 'Publications']

publication_counts_df.tail()
```

```
    Year  Publications
27  2020           841
28  2021           960
29  2022          1006
30  2023           974
31  2024            41
```

```python
# Using numpy for polynomial fitting
from numpy import polyfit, polyval
```

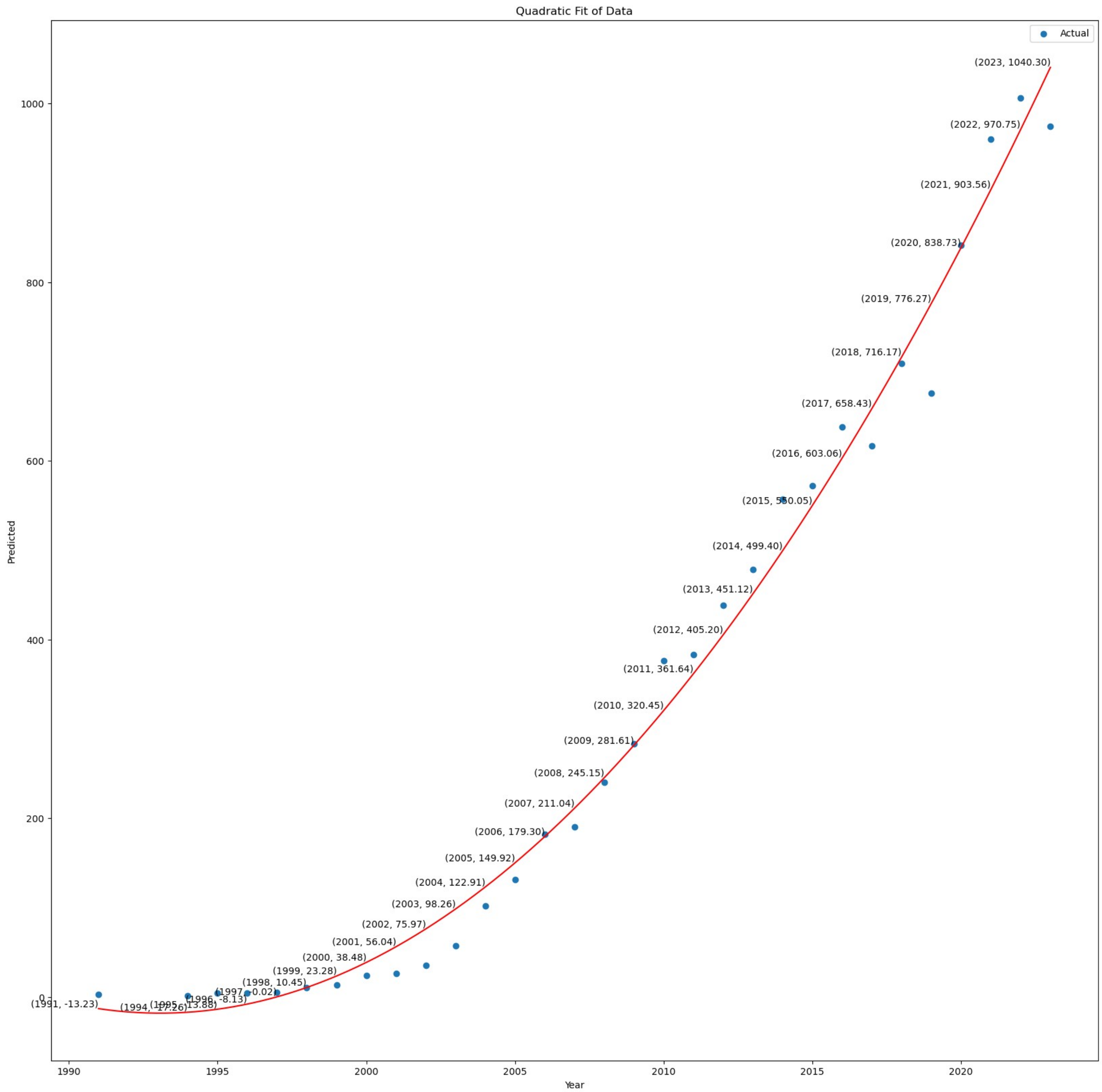
```python
# Filter data for pre-pandemic years for modeling
pre_pandemic_data =
publication_counts_df[publication_counts_df['Year'] <= 2019]

# Prepare data for modeling
years = pre_pandemic_data['Year'].values
publications = pre_pandemic_data['Publications'].values

# Polynomial degree (quadratic model, degree=2)
degree = 2

# Fit the model
publication_model = polyfit(years, publications, degree)

# Years for forecasting
forecast_years = np.array([1991, 1994, 1995, 1996, 1997, 1998, 1999,
2000, 2001, 2002,
                           2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010,
2011, 2012,
                           2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020,
2021, 2022,
                           2023])

# Forecasting
publication_forecasts = polyval(publication_model, forecast_years)
# Ensure no negative values in the forecast
publication_forecasts = np.maximum(publication_forecasts, 0)

# Actual data for comparison (we include 2020 since it's part of the
forecast now)
actual_publications =
publication_counts_df[publication_counts_df['Year'].isin(forecast_year
s)]['Publications'].values

# Prepare a DataFrame for manuscript-ready table
forecast_vs_actual = pd.DataFrame({
    'Year': forecast_years,
    'Forecasted Publications': publication_forecasts.round(0),
    'Actual Publications': actual_publications
})

forecast_vs_actual
```

	Year	Forecasted Publications	Actual Publications
0	1991	0.0	3
1	1994	0.0	1
2	1995	0.0	4
3	1996	0.0	4
4	1997	0.0	5
5	1998	11.0	10

6	1999	24.0	13
7	2000	39.0	24
8	2001	57.0	26
9	2002	76.0	35
10	2003	99.0	57
11	2004	123.0	102
12	2005	150.0	131
13	2006	179.0	182
14	2007	211.0	190
15	2008	245.0	240
16	2009	281.0	283
17	2010	319.0	376
18	2011	360.0	383
19	2012	403.0	438
20	2013	449.0	478
21	2014	497.0	557
22	2015	547.0	572
23	2016	599.0	638
24	2017	654.0	617
25	2018	711.0	709
26	2019	771.0	676
27	2020	833.0	841
28	2021	897.0	960
29	2022	964.0	1006
30	2023	1032.0	974

```
import matplotlib.pyplot as plt
import numpy as np
from scipy.optimize import curve_fit

# Data
years = np.array([1991, 1994, 1995, 1996, 1997, 1998, 1999, 2000,
2001, 2002,
                   2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010,
2011, 2012,
                   2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020,
2021, 2022,
                   2023])
actual = np.array([3, 1, 4, 4, 5, 10, 13, 24, 26, 35, 57, 102, 131,
182, 190,
                    240, 283, 376, 383, 438, 478, 557, 572, 638, 617,
709, 676,
                    841, 960, 1006, 974])
# Quadratic function
def quadratic_func(x, a, b, c):
    return a * x**2 + b * x + c
# Fit a quadratic curve
popt, _ = curve_fit(quadratic_func, years, actual)
a, b, c = popt
```

```python
# Set the figure size
plt.figure(figsize=(20,20))

# Plot the data points
plt.scatter(years, actual, label='Actual')

# Plot the quadratic curve
x_vals = np.linspace(years[0], years[-1], 100)
y_vals = quadratic_func(x_vals, a, b, c)
plt.plot(x_vals, y_vals, 'r')

# Add labels for the predicted values
predicted_values = quadratic_func(years, a, b, c)
for i, year in enumerate(years):
    plt.text(year, predicted_values[i], f'({year},
{predicted_values[i]:.2f})', verticalalignment='bottom',
horizontalalignment='right')

# Add labels and legend
plt.xlabel('Year')
plt.ylabel('Predicted')
plt.title('Quadratic Fit of Data')
plt.legend()

# Show plot
plt.show()
```




```python
from sklearn.metrics import mean_squared_error, r2_score

# Calculate RMSE
rmse = np.sqrt(mean_squared_error(actual, predicted_values))

# Calculate R-squared
r_squared = r2_score(actual, predicted_values)

print(f"RMSE: {rmse:.2f}")
print(f"R-squared: {r_squared:.2f}")

RMSE: 35.00
R-squared: 0.99
```

```python
# Import necessary libraries
import pandas as pd
import statsmodels.api as sm
from sklearn.metrics import mean_squared_error, r2_score

# Load the data from the Excel file
path = 'Final Data CA+FA.csv'
data = pd.read_csv(path)

# Aggregate publication counts by year
publication_counts = data.groupby('Publication Year')['WOS
ID'].nunique()
publication_counts_df = publication_counts.reset_index()
publication_counts_df.columns = ['Year', 'Publications']

# Prepare features and target variable
X = publication_counts_df['Year'].values.reshape(-1, 1)
y = publication_counts_df['Publications'].values

# Add a constant for the intercept term in OLS
X_with_constant = sm.add_constant(X)

# Fit the OLS model
ols_model = sm.OLS(y, X_with_constant).fit()

# Make predictions
y_pred = ols_model.predict(X_with_constant)

# Calculate RMSE and R²
rmse = mean_squared_error(y, y_pred, squared=False)
r2 = r2_score(y, y_pred)

# Print the results
print(f"OLS Model RMSE: {rmse:.4f}")
print(f"OLS Model R²: {r2:.4f}")
```

```
OLS Model RMSE: 170.9140
OLS Model R²: 0.7274
```