In [ ]:
```
import pandas as pd
import numpy as np
import datetime as dt
```

In [ ]:
```
SELECTED_STATE_FIPS = 6
NORMALIZATION_FACTOR = 100000
NORMALIZATION_FLAG = False
MARKER_SIZE=5
PREFIX_PATH = '../team_work/data/'
FORECAST_IN_DAYS = 10
```

In [ ]:
```
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from math import sqrt

class MyRegressionModel:
    """
    df: should be in format <index = Datetime> <columns = state_name> <values = num
    """
    def __init__(self, x_train, y_train, x_test, y_test, fit_type='Linear', error_t
        self.degree = degree
        self.fit_type = fit_type
        self.error_type = error_type
        self.degree = degree
        x_train = x_train.reshape(-1,1)
        x_test = x_test.reshape(-1,1)
        if self.fit_type == 'Polynomial':
            poly = PolynomialFeatures(degree=self.degree)
            x_train = poly.fit_transform(x_train)
            x_test = poly.fit_transform(x_test)
        self.x_train = x_train
        self.y_train = y_train
        self.x_test = x_test
        self.y_test = y_test
        self.fit_data(x_train, y_train)
        self.test_model(x_test, y_test)

    def fit_data(self, x_train, y_train):
        pr = LinearRegression()
        x = x_train
        y = y_train
        pr.fit(x, y)
        self.model = pr


    def test_model(self, x_test, y_test):
        y_pred = self.model.predict(x_test).reshape(len(x_test))
        if self.error_type == 'RMSE':
            self.error = sqrt(abs(mean_squared_error(y_test, y_pred)))
        residuals_lr = self.y_train - self.model.predict(self.x_train)
        ci_lr = np.quantile(residuals_lr, 0.95)
        self.confidence = self.generate_confidence_interval(y_pred, ci_lr)
```

```
        print(self.confidence)

    def generate_confidence_interval(self, preds, ci):
        df = pd.DataFrame()
        df['prediction'] = preds
        if ci >= 0:
            df['upper'] = preds + ci
            df['lower'] = preds - ci
        else:
            df['upper'] = preds - ci
            df['lower'] = preds + ci
        return df

    def predict(self, x):
        if self.fit_type == 'Polynomial':
            poly = PolynomialFeatures(degree=self.degree)
            x = poly.fit_transform(x)
        return self.model.predict(x)




def get_county_population(county_name):
    df = pd.read_csv(f'{PREFIX_PATH}covid_county_population_usafacts.csv')
    df = df[df['County Name'] == county_name]
    return df['population'].values[0]

def get_state_population(state):
    """
    Returns the population of the selected state

    Args:
        state (str): State abb

    Returns:
        int: Population
    """
    population = pd.read_csv(f'{PREFIX_PATH}/covid_county_population_usafacts.csv')
    population = population.groupby('State').sum(numeric_only=True)
    return population.loc[state, 'population']
```

In [ ]:
```
covid_confirmed_df = pd.read_csv(f'{PREFIX_PATH}covid_confirmed_usafacts.csv')
covid_confirmed_df = covid_confirmed_df[covid_confirmed_df['countyFIPS'] != 0]
covid_confirmed_df = covid_confirmed_df[covid_confirmed_df['StateFIPS'] == SELECTED
```

In [ ]:
```
covid_deaths_df = pd.read_csv(f'{PREFIX_PATH}covid_deaths_usafacts.csv')
covid_deaths_df = covid_deaths_df[covid_deaths_df['countyFIPS'] != 0]
covid_deaths_df = covid_deaths_df[covid_deaths_df['StateFIPS'] == SELECTED_STATE_FI
```

In [ ]:
```
selected_state_name = covid_confirmed_df['State'].iloc[0]
covid_confirmed_df = covid_confirmed_df.drop(['State', 'StateFIPS'], axis=1)
covid_deaths_df = covid_deaths_df.drop(['State', 'StateFIPS'], axis=1)
```

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

Out[ ]:

	countyFIPS	County Name	2020-01-22	2020-01-23	2020-01-24	2020-01-25	2020-01-26	2020-01-27	2020-01-28	2020-01-29	...	2023-01-07
191	6001	Alameda County	4	4	4	5	5	5	5	5	...	369865
192	6003	Alpine County	0	0	0	0	0	0	0	0	...	138
193	6005	Amador County	1	1	1	1	1	1	1	1	...	10339

3 rows × 1093 columns

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

Out[ ]:

	countyFIPS	County Name	2020-01-22	2020-01-23	2020-01-24	2020-01-25	2020-01-26	2020-01-27	2020-01-28	2020-01-29	...	2023-01-07	2 0
191	6001	Alameda County	0	0	0	0	0	0	0	0	...	2098	
192	6003	Alpine County	0	0	0	0	0	0	0	0	...	0	
193	6005	Amador County	0	0	0	0	0	0	0	0	...	96	

3 rows × 1093 columns

In [ ]:
```
covid_county_most_cases = covid_confirmed_df.sort_values(by=str(covid_confirmed_df.
covid_county_most_cases = covid_county_most_cases.iloc[:5]
selected_counties_most_cases_fips = covid_county_most_cases['countyFIPS'].values
selected_counties_most_cases_names = covid_county_most_cases['County Name'].values
covid_county_most_cases = covid_county_most_cases.drop(['countyFIPS', 'County Name'
covid_county_most_cases
```

Out[ ]:

	2020-01-22	2020-01-23	2020-01-24	2020-01-25	2020-01-26	2020-01-27	2020-01-28	2020-01-29	2020-01-30	2020-01-31	...	2023-01-07	202 01-0
209	375	379	382	384	385	388	390	391	392	401	...	3433929	343392
227	0	0	0	0	0	0	0	0	0	0	...	967383	96738
223	60	60	60	60	60	61	61	61	61	62	...	723518	72351
226	76	78	78	80	81	83	84	85	86	89	...	699474	69947
220	8	8	9	9	9	9	9	9	10	10	...	697388	69738

5 rows × 1091 columns

In [ ]:
```
covid_county_most_deaths = covid_deaths_df.sort_values(by=str(covid_deaths_df.colum
covid_county_most_deaths = covid_county_most_deaths.iloc[:5]
selected_counties_most_deaths_fips = covid_county_most_deaths['countyFIPS']
selected_counties_most_deaths_names = covid_county_most_deaths['County Name']
covid_county_most_deaths = covid_county_most_deaths.drop(['countyFIPS', 'County Nam
covid_county_most_deaths
```

Out[ ]:

	2020-01-22	2020-01-23	2020-01-24	2020-01-25	2020-01-26	2020-01-27	2020-01-28	2020-01-29	2020-01-30	2020-01-31	...	2023-01-07	2023-01-08	2
209	0	0	0	0	0	0	0	0	0	0	...	34488	34488	3
226	0	0	0	0	0	0	0	0	0	0	...	8104	8104	
220	0	0	0	0	0	0	0	0	0	0	...	7683	7683	
223	0	0	0	0	0	0	0	0	0	0	...	6714	6714	
227	0	0	0	0	0	0	0	0	0	0	...	5662	5662	

5 rows × 1091 columns

In [ ]:
```
covid_county_most_cases.columns = pd.to_datetime(covid_county_most_cases.columns)
covid_county_most_cases.index = selected_counties_most_cases_names
first_day_cases = covid_county_most_cases.iloc(1)[0]
covid_county_most_cases = covid_county_most_cases.diff(axis=1)
covid_county_most_cases = covid_county_most_cases.drop(covid_county_most_cases.colu
covid_county_most_cases = covid_county_most_cases.transpose()
covid_county_most_cases = covid_county_most_cases[covid_county_most_cases.select_dt
covid_county_most_cases.head(3)
```

Out[ ]:

	Los Angeles County	San Diego County	Riverside County	San Bernardino County	Orange County
2020-01-23	4	0	0	2	0
2020-01-24	3	0	0	0	1
2020-01-25	2	0	0	2	0

In [ ]:
```
covid_county_most_deaths.columns = pd.to_datetime(covid_county_most_deaths.columns)
covid_county_most_deaths.index = selected_counties_most_deaths_names
first_day_cases = covid_county_most_deaths.iloc(1)[0]
covid_county_most_deaths = covid_county_most_deaths.diff(axis=1)
covid_county_most_deaths = covid_county_most_deaths.drop(covid_county_most_deaths.c
covid_county_most_deaths = covid_county_most_deaths.transpose()
covid_county_most_deaths = covid_county_most_deaths[covid_county_most_deaths.select
covid_county_most_deaths.head(3)
```

Out[ ]:

County Name	Los Angeles County	San Bernardino County	Orange County	Riverside County	San Diego County
2020-01-23	0	0	0	0	0
2020-01-24	0	0	0	0	0
2020-01-25	0	0	0	0	0

In [ ]:
```
# For normalization
if NORMALIZATION_FLAG:
    for county_name in selected_counties_most_cases_names:
        covid_county_most_cases[county_name] = (covid_county_most_cases[county_name
    for county_name in selected_counties_most_deaths_names:
        covid_county_most_deaths[county_name] = (covid_county_most_deaths[county_na
```

In [ ]:
```
covid_county_1_most_cases = covid_county_most_cases[covid_county_most_cases.columns
covid_county_2_most_cases = covid_county_most_cases[covid_county_most_cases.columns
covid_county_3_most_cases = covid_county_most_cases[covid_county_most_cases.columns
covid_county_4_most_cases = covid_county_most_cases[covid_county_most_cases.columns
covid_county_5_most_cases = covid_county_most_cases[covid_county_most_cases.columns
covid_county_1_most_cases.head(3)
```

Out[ ]:

	Los Angeles County
0	4
1	3
2	2

In [ ]:
```
covid_county_1_most_deaths = covid_county_most_deaths[covid_county_most_deaths.colu
covid_county_2_most_deaths = covid_county_most_deaths[covid_county_most_deaths.colu
covid_county_3_most_deaths = covid_county_most_deaths[covid_county_most_deaths.colu
covid_county_4_most_deaths = covid_county_most_deaths[covid_county_most_deaths.colu
covid_county_5_most_deaths = covid_county_most_deaths[covid_county_most_deaths.colu
covid_county_1_most_deaths.head(3)
```

Out[ ]:

	Los Angeles County
0	0
1	0
2	0

In [ ]:
```
covid_county_most_cases = covid_county_most_cases.reset_index().drop(['index'], axi
covid_county_most_deaths = covid_county_most_deaths.reset_index().drop(['index'], a
covid_county_most_cases.head(3)
```

Out[ ]:

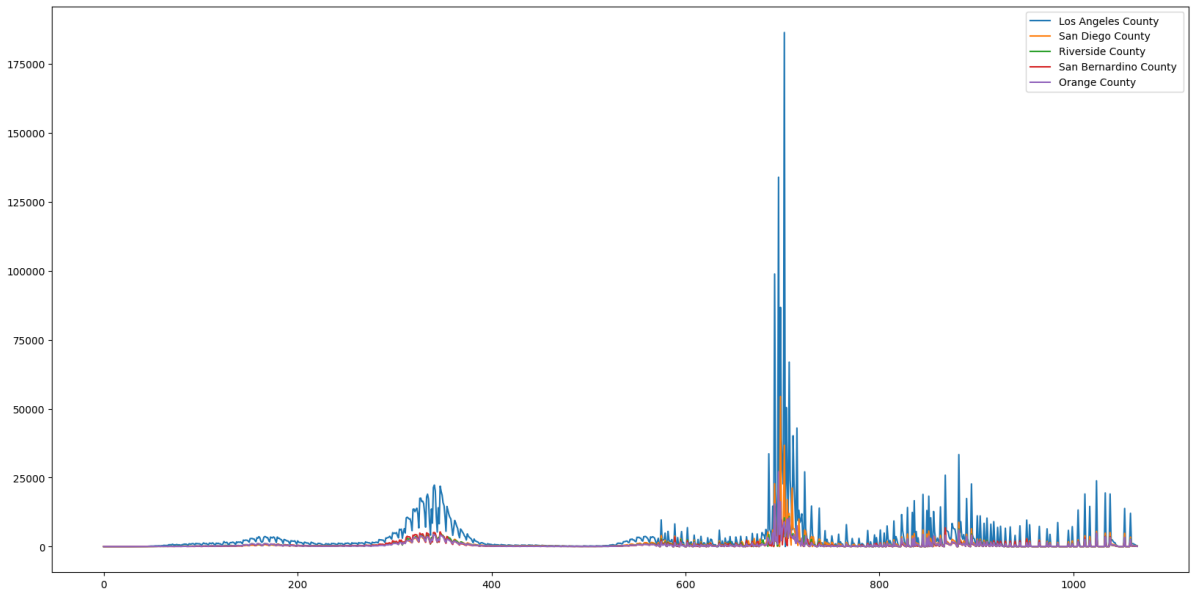
	Los Angeles County	San Diego County	Riverside County	San Bernardino County	Orange County
0	4	0	0	2	0
1	3	0	0	0	1
2	2	0	0	2	0

In [ ]:
```
covid_state_most_cases = covid_county_most_cases.sum(axis=1)
covid_state_most_deaths = covid_county_most_deaths.sum(axis=1)
covid_state_most_cases.head(3)
```

Out[ ]:
```
0    6
1    4
2    4
dtype: int64
```

In [ ]:
```
covid_county_most_cases.plot(figsize=(20,10))
```

Out[ ]:
```
<Axes: >
```



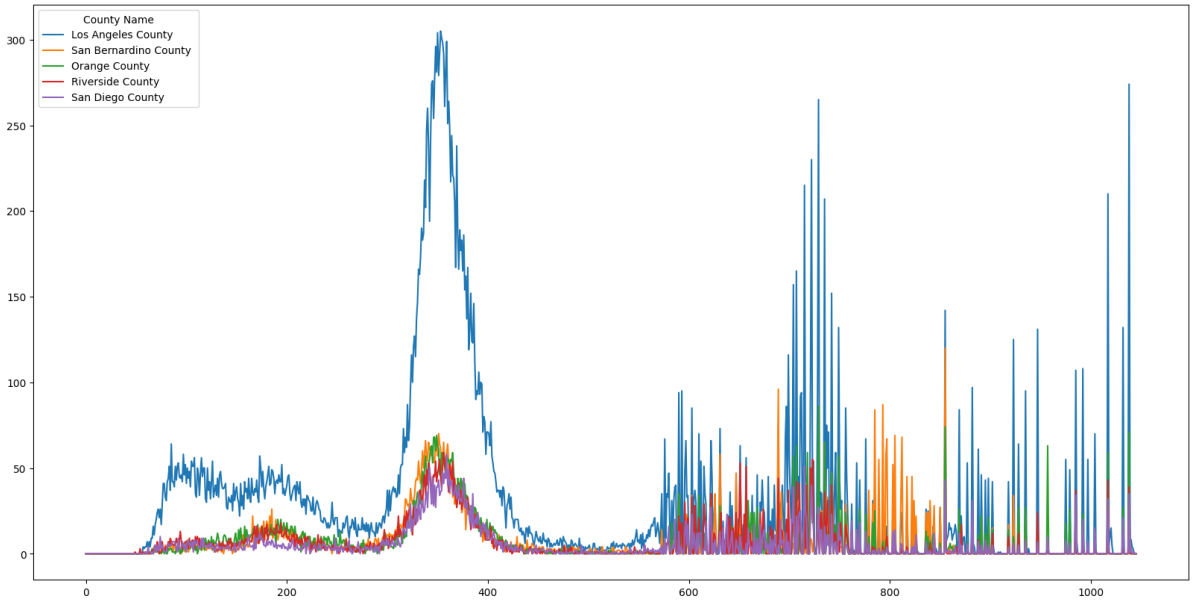
Analysis

The above plot shows the comparison between the top 5 counties in the selected state in terms of new covid-19 cases along a specific timeframe

We can see that Los Angelos comes at first place with the highest number of new cases along the whole timeframe.

In [ ]:
```
covid_county_most_deaths.plot(figsize=(20,10))
```

Out[ ]:
```
<Axes: >
```



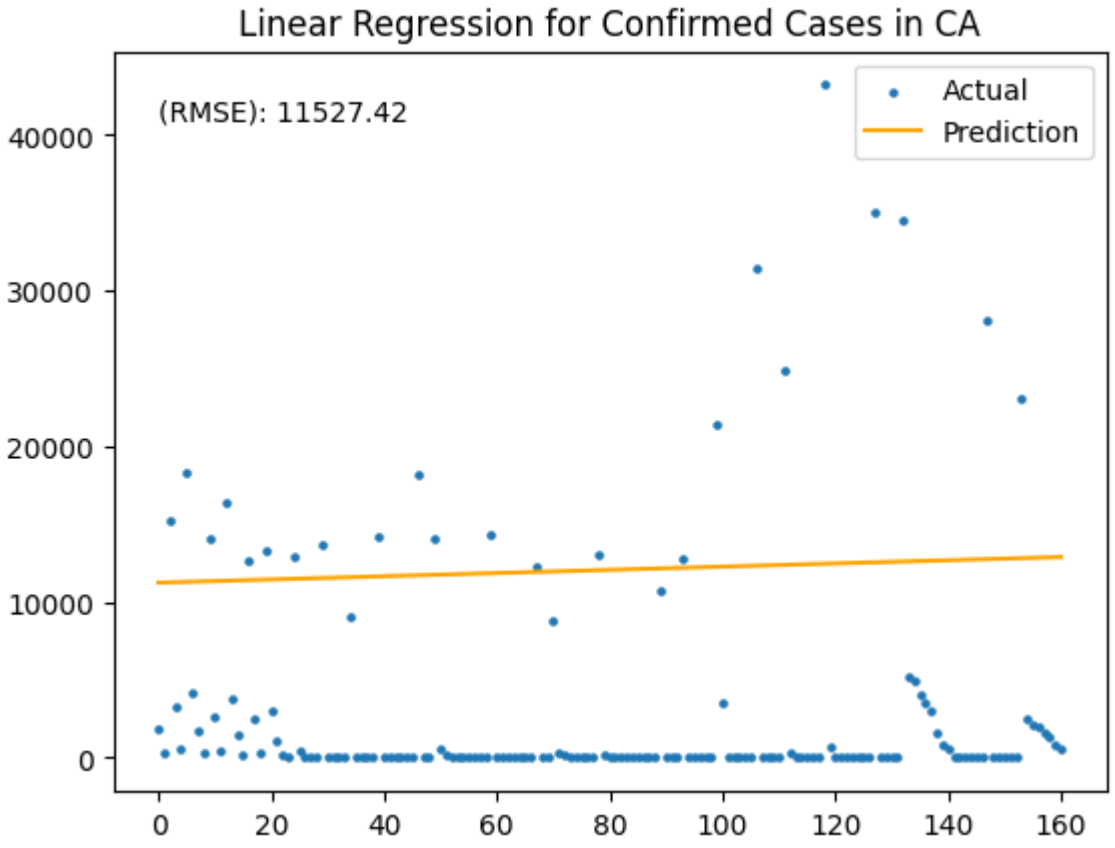
In [ ]:
```
from sklearn import preprocessing, svm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.model_selection import train_test_split
from math import sqrt
import matplotlib.pyplot as plt
```

In [ ]:
```
x_cases = np.array(covid_state_most_cases.index).reshape(-1,1).astype(int)
y_cases = covid_state_most_cases.values.astype(int)
x_cases_train, x_cases_test, y_cases_train, y_cases_test = train_test_split(x_cases
lrl_cases = LinearRegression()
lrl_cases.fit(x_cases_train, y_cases_train)
y_pred_lrl_cases = lrl_cases.predict(x_cases_test)
rmse_lrl_cases = round(sqrt( mean_squared_error(y_cases_test, y_pred_lrl_cases)),2)
print(f'Linear regression Root Mean Square Error (RMSE): {rmse_lrl_cases}')
```

```
Linear regression Root Mean Square Error (RMSE): 11527.42
```

In [ ]:
```
plt.text(x=0,y=41000,s=f'(RMSE): {rmse_lrl_cases}')
plt.scatter(np.arange(len(y_cases_test)),y_cases_test, label='Actual', s=MARKER_SIZ
plt.plot(y_pred_lrl_cases, color='orange', label='Prediction')
plt.legend()
plt.title(f'Linear Regression for Confirmed Cases in {selected_state_name}')
```

Out[ ]:
```
Text(0.5, 1.0, 'Linear Regression for Confirmed Cases in CA')
```



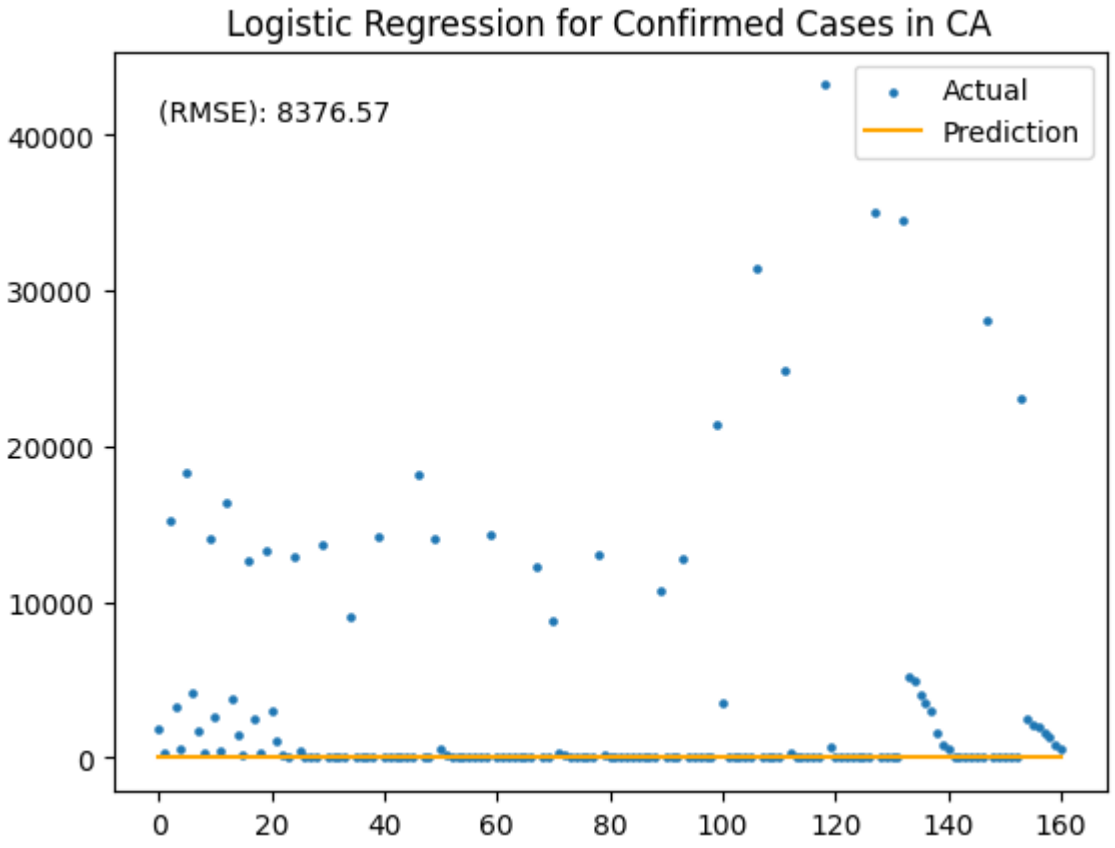
In [ ]:
```
lrg_cases = LogisticRegression()
lrg_cases.fit(x_cases_train,y_cases_train)
y_pred_cases_lrg = lrg_cases.predict(x_cases_test)
rmse_cases_lrg = round(sqrt( mean_squared_error(y_cases_test, y_pred_cases_lrg)),2)
print(f'Logestic regression Root Mean Square Error (RMSE): {rmse_cases_lrg}')
```

```
Logestic regression Root Mean Square Error (RMSE): 8376.57
c:\Users\mosta\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\l
inear_model\_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

In [ ]:
```
plt.title(f'Logistic Regression for Confirmed Cases in {selected_state_name}')
plt.text(x=0,y=41000,s=f'(RMSE): {rmse_cases_lrg}')
plt.scatter(np.arange(len(y_cases_test)),y_cases_test,label='Actual',s=MARKER_SIZE)
plt.plot(y_pred_cases_lrg, color='orange', label='Prediction', scaley='log')
plt.legend()
```

Out[ ]:
```
<matplotlib.legend.Legend at 0x1f10d5a0090>
```



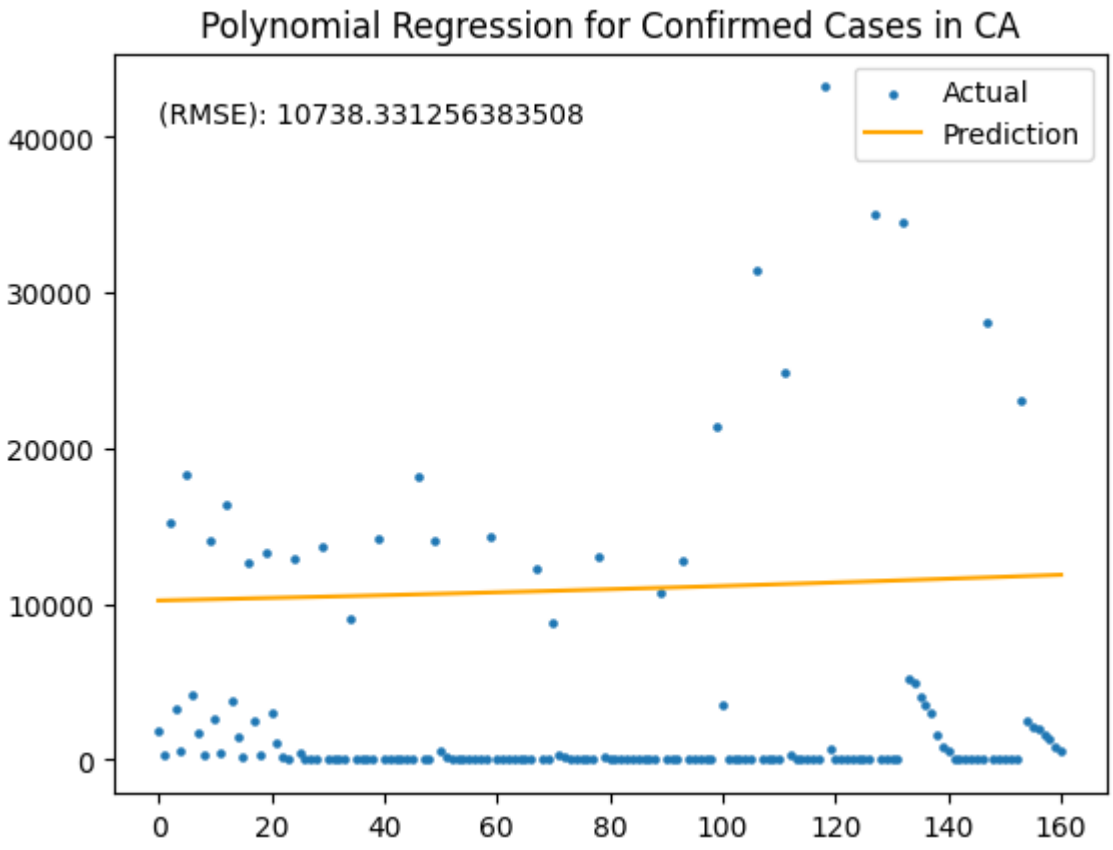
In [ ]:
```
model_1 = MyRegressionModel(x_cases_train, y_cases_train, x_cases_test, y_cases_tes
```

	prediction	upper	lower
0	10233.316425	30996.166551	-10529.533701
1	10241.279906	31004.130032	-10521.570220
2	10249.267400	31012.117526	-10513.582727
3	10257.279013	31020.129139	-10505.571114
4	10265.314852	31028.164978	-10497.535274
..	...	...	...
156	11832.226156	32595.076283	-8930.623970
157	11845.227049	32608.077176	-8917.623077
158	11858.268619	32621.118745	-8904.581507
159	11871.350972	32634.201098	-8891.499155
160	11884.474214	32647.324341	-8878.375912

[161 rows x 3 columns]

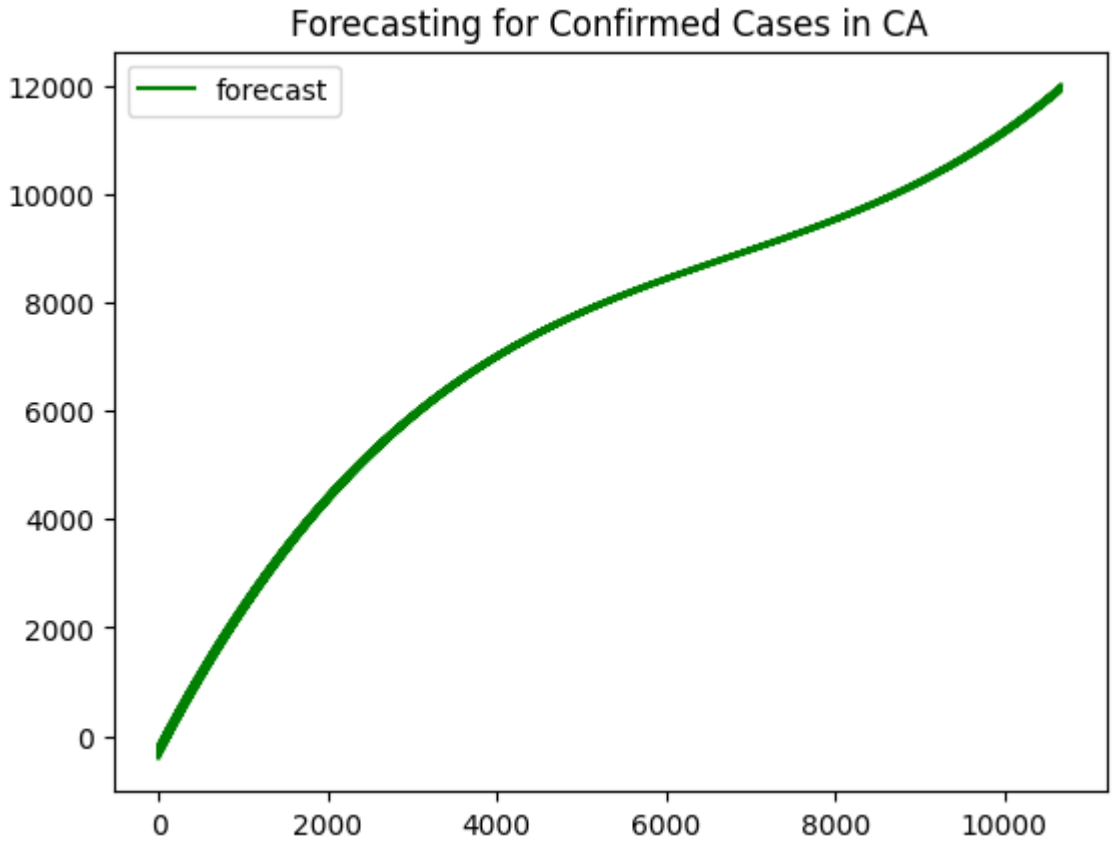
In [ ]:
```
plt.title(f'Polynomial Regression for Confirmed Cases in {selected_state_name}')
plt.text(x=0,y=41000,s=f'(RMSE): {model_1.error}')
plt.scatter(np.arange(len(y_cases_test)),y_cases_test,label='Actual',s=MARKER_SIZE)
plt.plot(model_1.predict(x_cases_test), color='orange', label='Prediction', scaley=
plt.legend()
```

Out[ ]:
```
<matplotlib.legend.Legend at 0x1f10d610d50>
```



In [ ]:
```
plt.title(f'Forecasting for Confirmed Cases in {selected_state_name}')
x_forecast = x_cases + [np.arange(FORECAST_IN_DAYS)]
plt.plot(model_1.predict(x_forecast.reshape(-1,1)), color='green', label='forecast'
```

Out[ ]:
```
<matplotlib.legend.Legend at 0x1f10d688450>
```

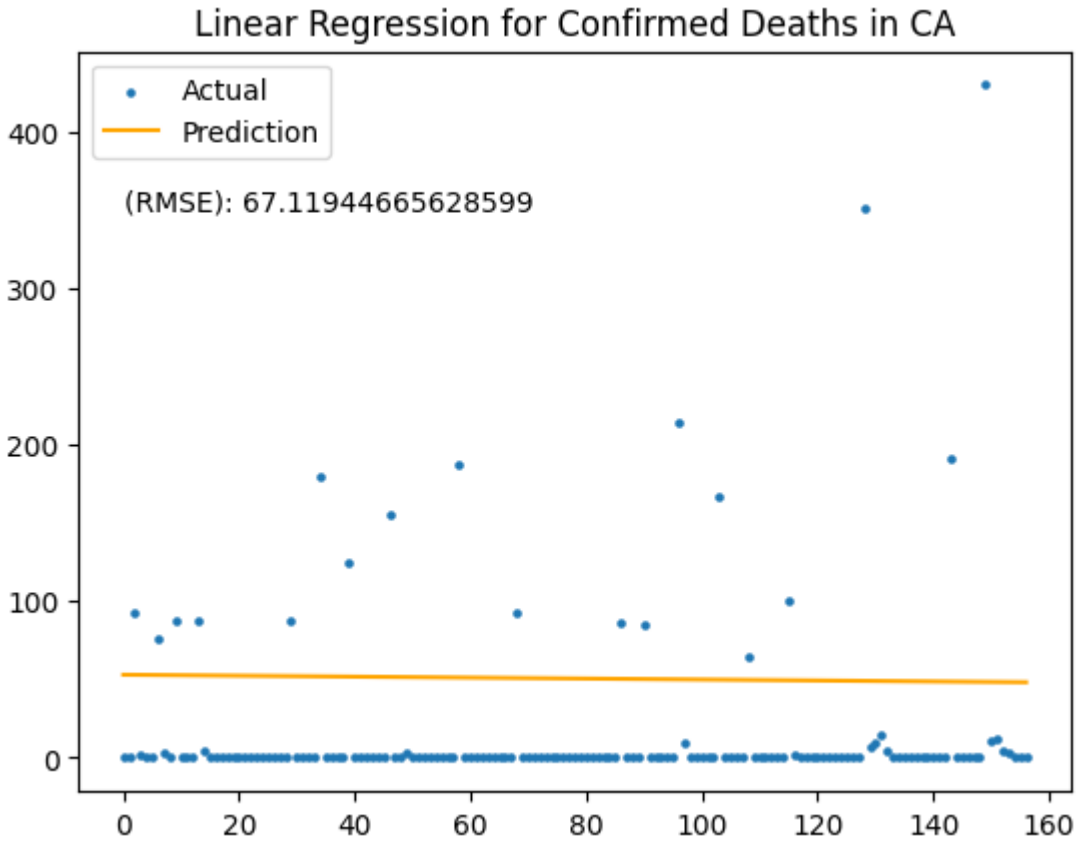


In [ ]:
```
x_deaths = np.array(covid_state_most_deaths.index).reshape(-1,1).astype(int)
y_deaths = covid_state_most_deaths.values.astype(int)
x_deaths_train, x_deaths_test, y_deaths_train, y_deaths_test = train_test_split(x_d
lrl_deaths = LinearRegression()
lrl_deaths.fit(x_deaths_train, y_deaths_train)
y_pred_lrl_deaths = lrl_deaths.predict(x_deaths_test)
rmse_lrl_deaths = sqrt( mean_squared_error(y_deaths_test, y_pred_lrl_deaths))
print(f'Linear regression Root Mean Square Error (RMSE): {round(rmse_lrl_deaths,2)}
```

```
Linear regression Root Mean Square Error (RMSE): 67.12
```

In [ ]:
```
plt.scatter(np.arange(len(y_deaths_test)),y_deaths_test, label='Actual',s=MARKER_SI
plt.plot(y_pred_lrl_deaths, color='orange', label='Prediction')
plt.legend()
plt.title('Linear Regression for Confirmed Deaths in ' + selected_state_name)
plt.text(x=0, y=350, s=f'(RMSE): {rmse_lrl_deaths}')
```

Out[ ]:
```
Text(0, 350, '(RMSE): 67.11944665628599')
```



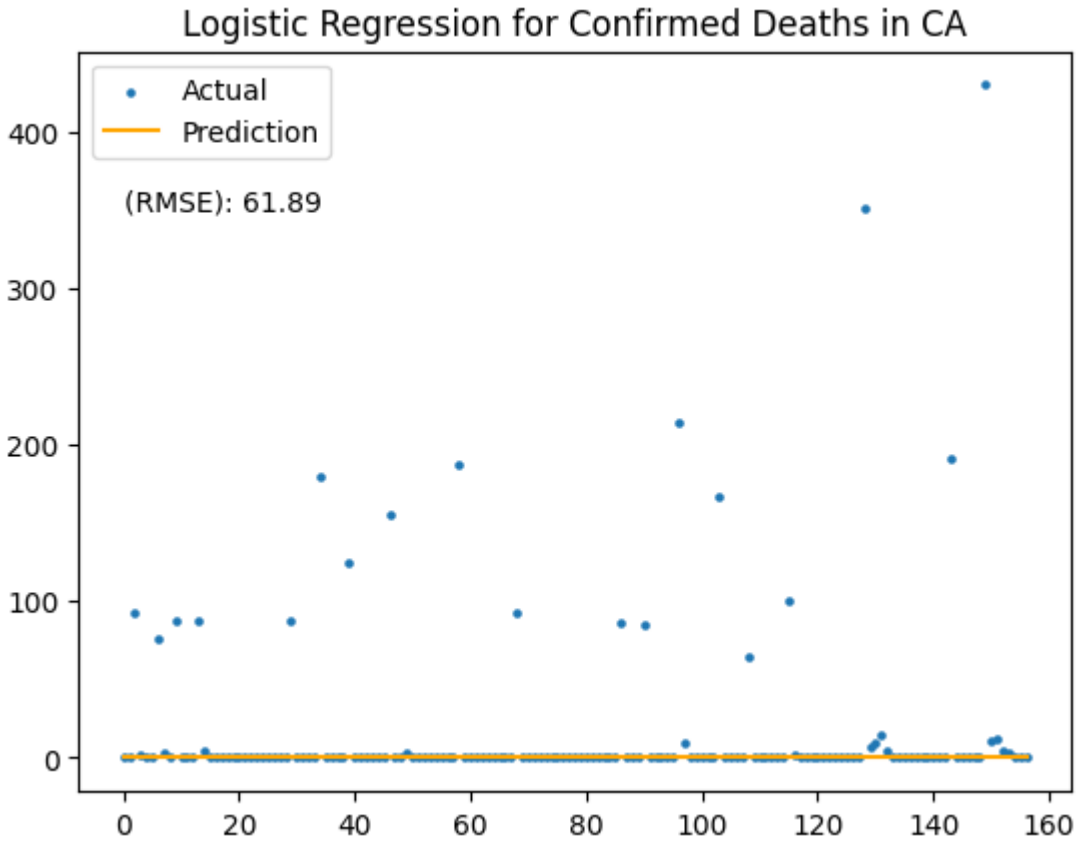
In [ ]:
```
lrg_deaths = LogisticRegression()
lrg_deaths.fit(x_deaths_train,y_deaths_train)
y_pred_deaths_lrg = lrg_deaths.predict(x_deaths_test)
rmse_deaths_lrg = round(sqrt( mean_squared_error(y_deaths_test, y_pred_deaths_lrg))
print(f'Logestic regression Root Mean Square Error (RMSE): {rmse_deaths_lrg}')
```

```
Logestic regression Root Mean Square Error (RMSE): 61.89
c:\Users\mosta\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\l
inear_model\_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status
=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

In [ ]:
```
plt.title(f'Logistic Regression for Confirmed Deaths in {selected_state_name}')
plt.text(x=0,y=350,s=f'(RMSE): {rmse_deaths_lrg}')
plt.scatter(np.arange(len(y_deaths_test)),y_deaths_test,label='Actual',s=MARKER_SIZ
plt.plot(y_pred_deaths_lrg, color='orange', label='Prediction', scaley='log')
plt.legend()
```

Out[ ]:
```
<matplotlib.legend.Legend at 0x1f10d510d50>
```



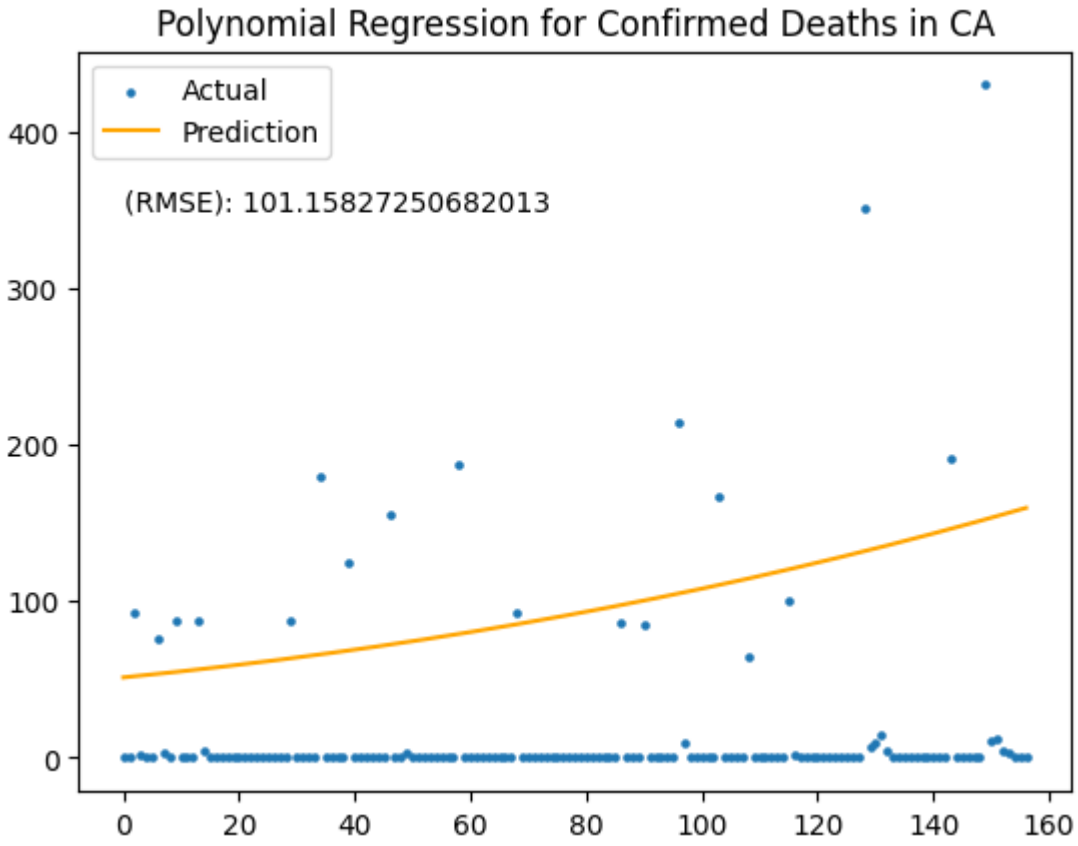
In [ ]:
```
model_2 = MyRegressionModel(x_deaths_train, y_deaths_train, x_deaths_test, y_deaths
```

	prediction	upper	lower
0	51.411253	276.110670	-173.288165
1	51.780375	276.479793	-172.919042
2	52.153165	276.852583	-172.546253
3	52.529632	277.229050	-172.169786
4	52.909786	277.609204	-171.789631
..	...	...	...
152	155.536573	380.235991	-69.162844
153	156.581817	381.281235	-68.117601
154	157.632301	382.331718	-67.067117
155	158.688035	383.387452	-66.011383
156	159.749029	384.448447	-64.950389

[157 rows x 3 columns]

In [ ]:
```
plt.scatter(np.arange(len(y_deaths_test)),y_deaths_test, s=MARKER_SIZE)
plt.plot(model_2.predict(x_deaths_test), color='orange')
plt.text(x=0,y=350,s=f'(RMSE): {model_2.error}')
plt.title(f'Polynomial Regression for Confirmed Deaths in {selected_state_name}')
plt.legend(['Actual', 'Prediction'])
```

Out[ ]:
```
<matplotlib.legend.Legend at 0x1f10a739650>
```



In [ ]:
```
x_county_1_cases = np.array(covid_county_1_most_cases.index).reshape(-1,1).astype(i
x_county_2_cases = np.array(covid_county_2_most_cases.index).reshape(-1,1).astype(i
x_county_3_cases = np.array(covid_county_3_most_cases.index).reshape(-1,1).astype(i
x_county_4_cases = np.array(covid_county_4_most_cases.index).reshape(-1,1).astype(i
x_county_5_cases = np.array(covid_county_5_most_cases.index).reshape(-1,1).astype(i
y_county_1_cases = covid_county_1_most_cases.values.astype(int)
y_county_2_cases = covid_county_2_most_cases.values.astype(int)
y_county_3_cases = covid_county_3_most_cases.values.astype(int)
y_county_4_cases = covid_county_4_most_cases.values.astype(int)
y_county_5_cases = covid_county_5_most_cases.values.astype(int)
```

In [ ]:
```
x_county_1_cases_train, x_county_1_cases_test, y_county_1_cases_train, y_county_1_c
x_county_2_cases_train, x_county_2_cases_test, y_county_2_cases_train, y_county_2_c
x_county_3_cases_train, x_county_3_cases_test, y_county_3_cases_train, y_county_3_c
x_county_4_cases_train, x_county_4_cases_test, y_county_4_cases_train, y_county_4_c
x_county_5_cases_train, x_county_5_cases_test, y_county_5_cases_train, y_county_5_c
```

In [ ]:
```
lrl_county_1_cases = LinearRegression()
lrl_county_2_cases = LinearRegression()
lrl_county_3_cases = LinearRegression()
lrl_county_4_cases = LinearRegression()
lrl_county_5_cases = LinearRegression()
```

In [ ]:
```
lrl_county_1_cases.fit(x_county_1_cases_train, y_county_1_cases_train)
lrl_county_2_cases.fit(x_county_2_cases_train, y_county_2_cases_train)
lrl_county_3_cases.fit(x_county_3_cases_train, y_county_3_cases_train)
lrl_county_4_cases.fit(x_county_4_cases_train, y_county_4_cases_train)
lrl_county_5_cases.fit(x_county_5_cases_train, y_county_5_cases_train)
```

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

In [ ]:
```
y_pred_lrl_county_1_cases = lrl_county_1_cases.predict(x_county_1_cases_test)
y_pred_lrl_county_2_cases = lrl_county_2_cases.predict(x_county_2_cases_test)
y_pred_lrl_county_3_cases = lrl_county_3_cases.predict(x_county_3_cases_test)
y_pred_lrl_county_4_cases = lrl_county_4_cases.predict(x_county_4_cases_test)
y_pred_lrl_county_5_cases = lrl_county_5_cases.predict(x_county_5_cases_test)
```

In [ ]:
```
rmse_lrl_county_1_cases = round(sqrt( mean_squared_error(y_county_1_cases_test, y_p
rmse_lrl_county_2_cases = round(sqrt( mean_squared_error(y_county_2_cases_test, y_p
rmse_lrl_county_3_cases = round(sqrt( mean_squared_error(y_county_3_cases_test, y_p
rmse_lrl_county_4_cases = round(sqrt( mean_squared_error(y_county_4_cases_test, y_p
rmse_lrl_county_5_cases = round(sqrt( mean_squared_error(y_county_5_cases_test, y_p
```

In [ ]:
```
print(f'Linear regression Root Mean Square Error (RMSE): {selected_counties_most_ca
print(f'Linear regression Root Mean Square Error (RMSE): {selected_counties_most_ca
print(f'Linear regression Root Mean Square Error (RMSE): {selected_counties_most_ca
print(f'Linear regression Root Mean Square Error (RMSE): {selected_counties_most_ca
print(f'Linear regression Root Mean Square Error (RMSE): {selected_counties_most_ca
```

```
Linear regression Root Mean Square Error (RMSE): Los Angeles County             6
357.82
Linear regression Root Mean Square Error (RMSE): San Diego County               1
852.23
Linear regression Root Mean Square Error (RMSE): Riverside County               1
122.13
Linear regression Root Mean Square Error (RMSE): San Bernardino County          1
101.61
Linear regression Root Mean Square Error (RMSE): Orange County                  1
156.27
```

In [ ]:
```
print(f'Confidence inteval for {selected_counties_most_cases_names[0]}')
model_3 = MyRegressionModel(x_county_1_cases_train, y_county_1_cases_train, x_count
print(f'Confidence inteval for {selected_counties_most_cases_names[1]}')
model_4 = MyRegressionModel(x_county_2_cases_train, y_county_2_cases_train, x_count
print(f'Confidence inteval for {selected_counties_most_cases_names[2]}')
model_5 = MyRegressionModel(x_county_3_cases_train, y_county_3_cases_train, x_count
print(f'Confidence inteval for {selected_counties_most_cases_names[3]}')
model_6 = MyRegressionModel(x_county_4_cases_train, y_county_4_cases_train, x_count
print(f'Confidence inteval for {selected_counties_most_cases_names[4]}')
model_7 = MyRegressionModel(x_county_5_cases_train, y_county_5_cases_train, x_count
```

```
Confidence inteval for Los Angeles County
      prediction         upper        lower
0    5516.090709  15980.126248 -4947.944831
1    5518.613980  15982.649519 -4945.421559
2    5521.135061  15985.170600 -4942.900479
3    5523.653968  15987.689507 -4940.381571
4    5526.170720  15990.206259 -4937.864819
..           ...           ...          ...
156  5894.336697  16358.372236 -4569.698843
157  5896.734505  16360.770044 -4567.301034
158  5899.132914  16363.168453 -4564.902625
159  5901.531941  16365.567480 -4562.503598
160  5903.931605  16367.967144 -4560.103935

[161 rows x 3 columns]
Confidence inteval for San Diego County
      prediction        upper        lower
0    1522.981895  4048.812253 -1002.848463
1    1521.666954  4047.497312 -1004.163404
2    1520.337899  4046.168257 -1005.492459
3    1518.994705  4044.825063 -1006.835653
4    1517.637347  4043.467705 -1008.193011
..           ...          ...          ...
156  1131.547912  3657.378270 -1394.282445
157  1127.726008  3653.556366 -1398.104350
158  1123.886051  3649.716409 -1401.944306
159  1120.028018  3645.858375 -1405.802340
160  1116.151881  3641.982239 -1409.678476

[161 rows x 3 columns]
Confidence inteval for Riverside County
      prediction        upper        lower
0     995.368582  3383.662005 -1392.924841
1     996.890518  3385.183941 -1391.402905
2     998.423157  3386.716580 -1389.870265
3     999.966534  3388.259957 -1388.326889
4    1001.520682  3389.814104 -1386.772741
..           ...          ...          ...
156  1383.127709  3771.421132 -1005.165713
157  1386.727151  3775.020574 -1001.566271
158  1390.342558  3778.635981  -997.950865
159  1393.973964  3782.267387  -994.319459
160  1397.621402  3785.914825  -990.672021

[161 rows x 3 columns]
Confidence inteval for San Bernardino County
      prediction        upper        lower
0    1033.597539  3349.231327 -1282.036248
1    1036.467827  3352.101615 -1279.165960
2    1039.356702  3354.990490 -1276.277085
3    1042.264216  3357.898004 -1273.369571
4    1045.190419  3360.824207 -1270.443368
..           ...          ...          ...
156  1737.704270  4053.338058  -577.929517
157  1744.090168  4059.723955  -571.543620
158  1750.502601  4066.136388  -565.131187
```

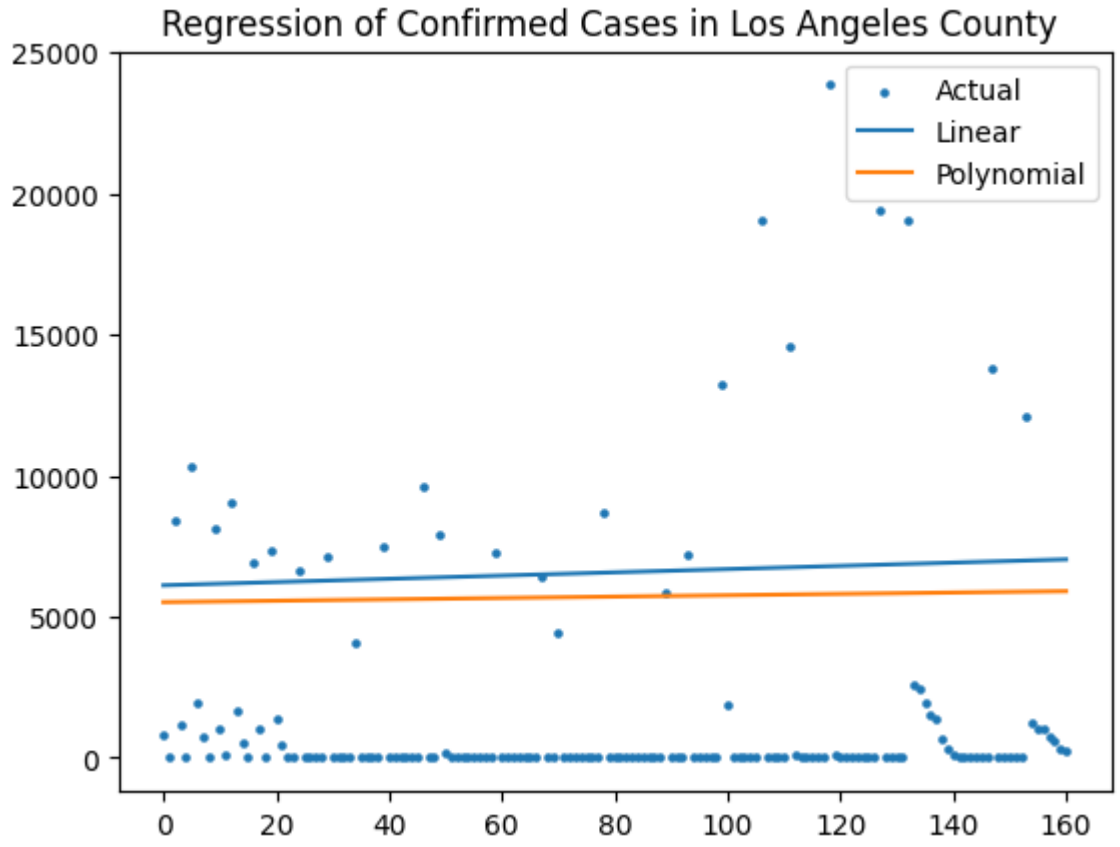
```
159  1756.941620  4072.575407  -558.692168
160  1763.407277  4079.041064  -552.226511

[161 rows x 3 columns]
Confidence inteval for Orange County
      prediction        upper        lower
0    1165.277701  3549.550607 -1218.995206
1    1167.640627  3551.913534 -1216.632279
2    1170.014581  3554.287487 -1214.258326
3    1172.399590  3556.672496 -1211.873317
4    1174.795685  3559.068591 -1209.477222
..           ...          ...          ...
156  1685.509568  4069.782474  -698.763338
157  1689.949218  4074.222124  -694.323689
158  1694.404495  4078.677402  -689.868411
159  1698.875430  4083.148336  -685.397477
160  1703.362051  4087.634957  -680.910856

[161 rows x 3 columns]
```

In [ ]:
```
plt.scatter(np.arange(len(y_county_1_cases_test)),y_county_1_cases_test, label=f'Ac
plt.plot(y_pred_lrl_county_1_cases, label=f'Linear')
plt.plot(model_3.predict(x_county_1_cases_test), label=f'Polynomial')
plt.legend()
plt.title(f'Regression of Confirmed Cases in {selected_counties_most_cases_names[0]
print(f'RMSE: {model_3.error}')
```

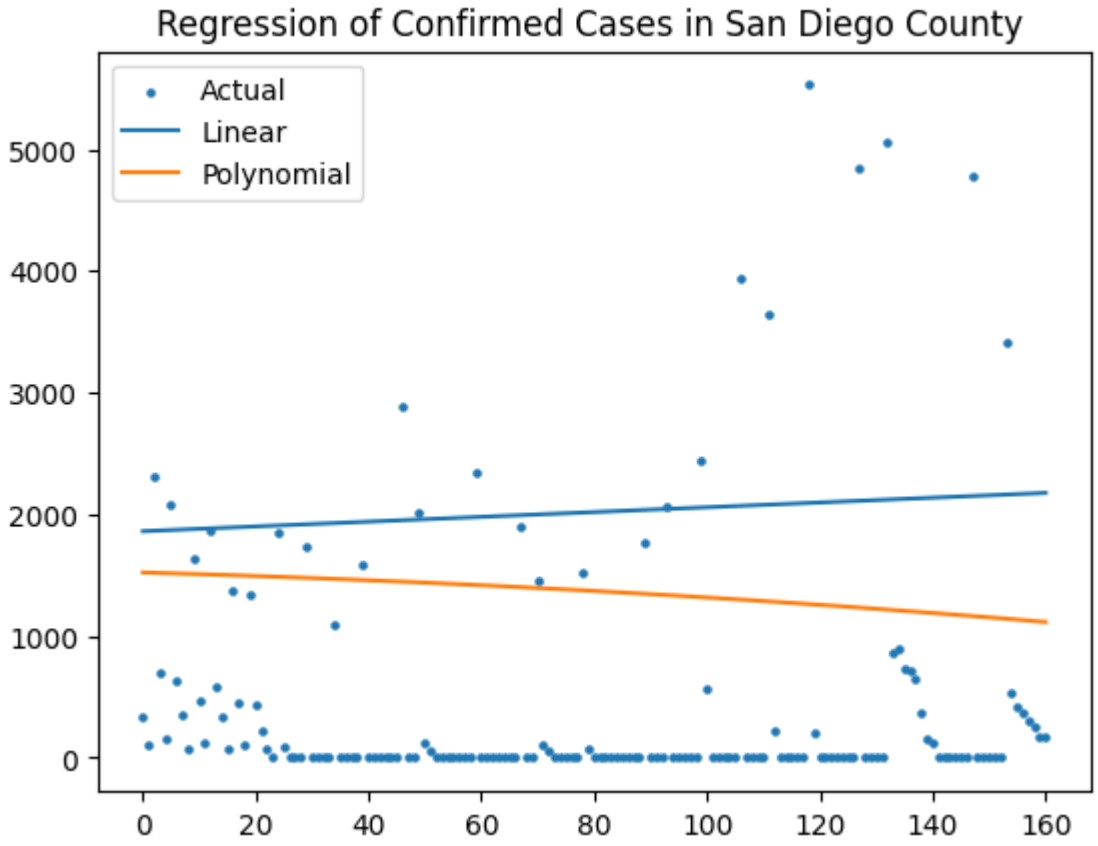
```
RMSE: 5745.198930058755
```



In [ ]:
```
plt.scatter(np.arange(len(y_county_2_cases_test)),y_county_2_cases_test, label=f'Ac
plt.plot(y_pred_lrl_county_2_cases, label=f'Linear')
```

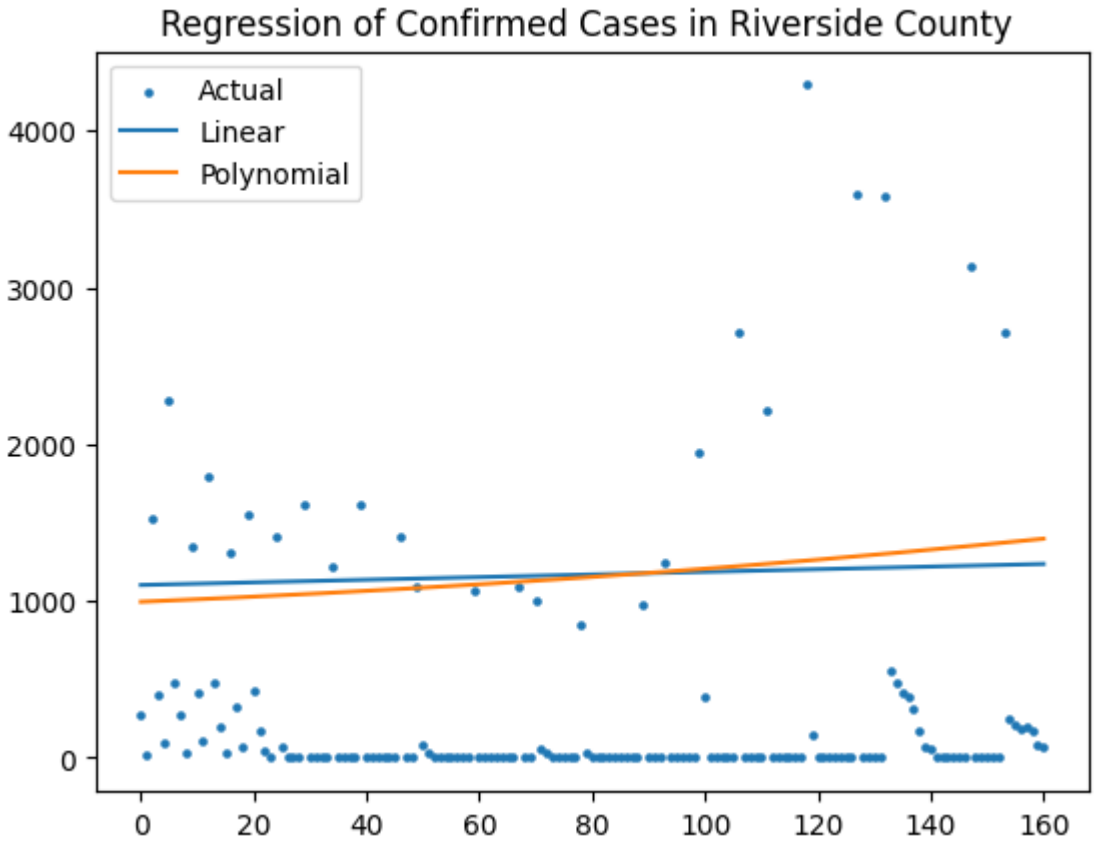
```
plt.plot(model_4.predict(x_county_2_cases_test), label=f'Polynomial')
plt.legend()
plt.title(f'Regression of Confirmed Cases in {selected_counties_most_cases_names[1]
print(f'RMSE: {model_4.error}')
```

```
RMSE: 1369.587334416514
```



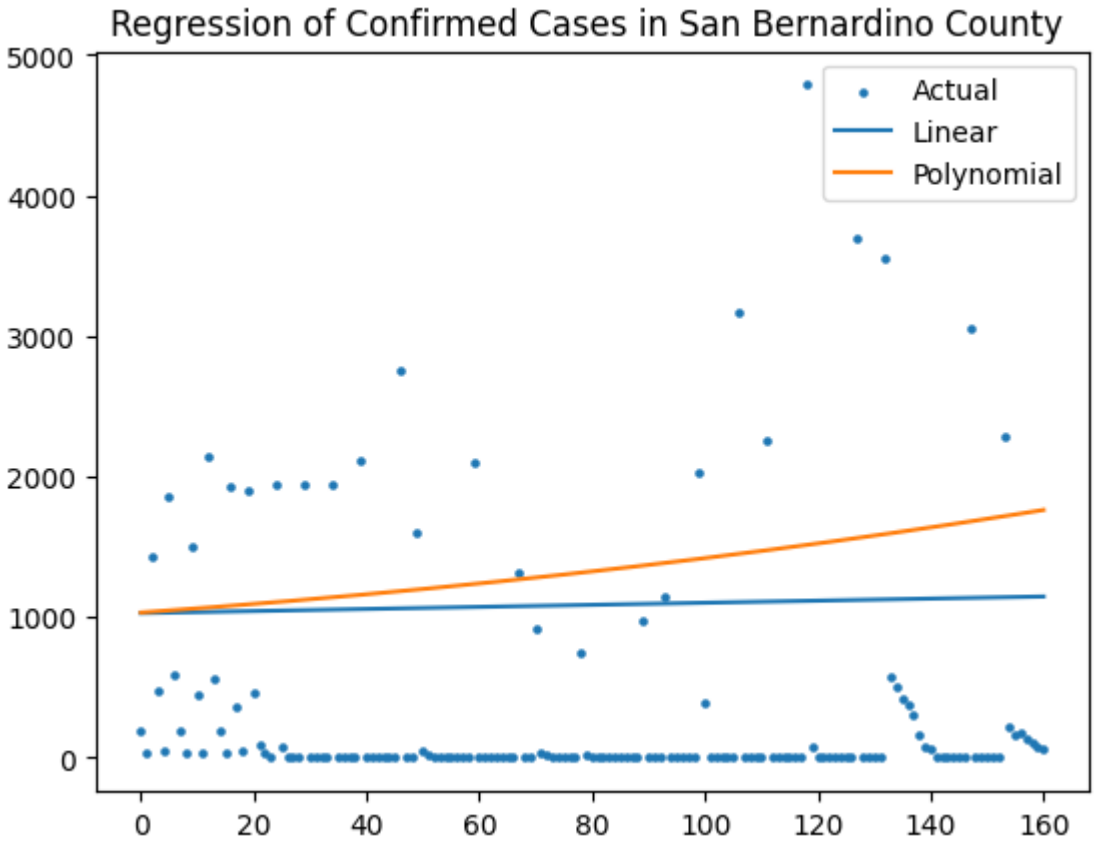
In [ ]:
```
plt.scatter(np.arange(len(y_county_3_cases_test)),y_county_3_cases_test, label=f'Ac
plt.plot(y_pred_lrl_county_3_cases, label=f'Linear')
plt.plot(model_5.predict(x_county_3_cases_test), label=f'Polynomial')
plt.legend()
plt.title(f'Regression of Confirmed Cases in {selected_counties_most_cases_names[2]
print(f'RMSE: {model_5.error}')
```

```
RMSE: 1125.4924045873065
```



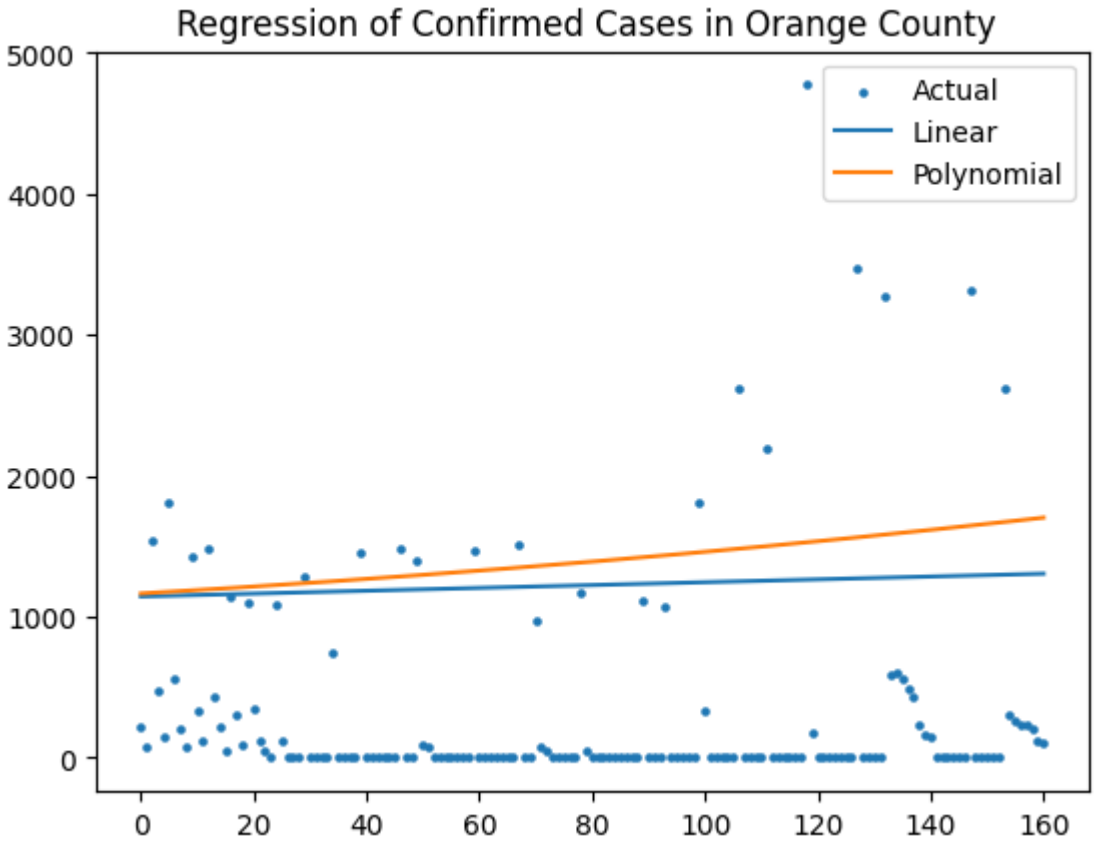
In [ ]:
```
plt.scatter(np.arange(len(y_county_4_cases_test)),y_county_4_cases_test, label=f'Ac
plt.plot(y_pred_lrl_county_4_cases, label=f'Linear')
plt.plot(model_6.predict(x_county_4_cases_test), label=f'Polynomial')
plt.legend()
plt.title(f'Regression of Confirmed Cases in {selected_counties_most_cases_names[3]
print(f'RMSE: {model_6.error}')
```

```
RMSE: 1303.8920457810668
```



In [ ]:
```
plt.scatter(np.arange(len(y_county_5_cases_test)),y_county_5_cases_test, label=f'Ac
plt.plot(y_pred_lrl_county_5_cases, label=f'Linear')
plt.plot(model_7.predict(x_county_5_cases_test), label=f'Polynomial')
plt.legend()
plt.title(f'Regression of Confirmed Cases in {selected_counties_most_cases_names[4]
print(f'RMSE: {model_7.error}')
```

```
RMSE: 1303.7670457212203
```



Observation

After doing the analysis and curve fitting on the counties mentioned above, we can see that for most of the counties numbers are on the rise, and even the forecasting of the models that we have says the same.

The most county at risk is Los Angelos, with the highest number in cases and deaths.

In [ ]:
```
enrichment_df = pd.read_csv(f'{PREFIX_PATH}enrichment.csv')
enrichment_df[enrichment_df.columns[3:]] = enrichment_df[enrichment_df.columns[3:]]
enrichment_df = enrichment_df[enrichment_df['Sex'] == 'T']
enrichment_df = enrichment_df.drop(['Sex', 'StateFIPS'], axis=1)
enrichment_df = enrichment_df.set_index(enrichment_df['State']).drop('State', axis=
enrichment_df = enrichment_df.sort_index()
enrichment_youth_df = enrichment_df[enrichment_df.columns[:8]]
enrichment_senior_df = enrichment_df[enrichment_df.columns[8:]]
enrichment_youth_df = enrichment_youth_df.aggregate(np.sum, axis=1)
enrichment_senior_df = enrichment_senior_df.aggregate(np.sum, axis=1)
enrichment_df.head(3)
```

Out[ ]:

State	Under 5 years	5 to 9 years	10 to 14 years	15 to 19 years	20 to 24 years	25 to 29 years	30 to 34 years	35 to 39 years	40 to 44 years	45 to 49 years	50 to 54 years	55 to 59 years	60 to 64 years	6 y
AK	6.3	7.1	7.2	6.9	6.3	7.5	7.9	7.5	6.6	5.5	5.8	5.9	6.4	
AL	5.8	5.9	6.7	6.7	6.4	6.2	6.3	6.2	6.5	6.0	6.3	6.6	6.8	
AR	5.9	6.5	6.8	6.8	6.6	6.1	6.6	6.5	6.2	5.8	6.0	6.5	6.3	

In [ ]:
```
covid_df = pd.read_csv(f'{PREFIX_PATH}covid_confirmed_usafacts.csv')
covid_df = covid_df[covid_df['countyFIPS'] != 0]
covid_df = covid_df.drop(['countyFIPS', 'County Name', 'StateFIPS'], axis=1)
covid_df = covid_df.groupby('State').sum()
covid_df = covid_df.transpose()
covid_df.index = pd.to_datetime(covid_df.index)
covid_df = covid_df.diff(axis=0)
covid_df = covid_df.drop(covid_df.index[0])
covid_df = covid_df.transpose()
covid_df = covid_df.sort_index()
covid_df = covid_df.groupby(covid_df.columns.isocalendar().week, axis=1).sum()
covid_df.head(3)
```

Out[ ]:

State	week 1	2	3	4	5	6	7	8	9	10	.
AK	10912.0	17935.0	16681.0	15817.0	10726.0	6468.0	4117.0	2724.0	2675.0	3185.0	.
AL	109490.0	76397.0	134831.0	50737.0	100140.0	27598.0	17695.0	12873.0	11682.0	9903.0	.
AR	61242.0	72275.0	67772.0	50398.0	33487.0	17345.0	10844.0	11038.0	6311.0	5483.0	.

3 rows × 53 columns

In [ ]:
```
states_pop = pd.read_csv(f'{PREFIX_PATH}covid_county_population_usafacts.csv')
states_pop = states_pop[states_pop['countyFIPS'] != 0].drop(['countyFIPS', 'County
states_pop.head(3)
```

Out[ ]:

State	population
AK	731545
AL	4903185
AR	3017804

In [ ]:
```
covid_mean_df = covid_df.mean(axis=1).round()
covid_mean_normalized_df = covid_mean_df.divide(states_pop['population']) * NORMALI
```

```
covid_mean_normalized_df = covid_mean_normalized_df.round()
covid_mean_normalized_df.head(3)
```

Out[ ]:
```
State
AK    720.0
AL    614.0
AR    597.0
dtype: float64
```

In [ ]:
```
covid_enrich_df = {'Cases': covid_mean_normalized_df, 'Youth':enrichment_youth_df,
covid_enrich_df = pd.concat(covid_enrich_df, axis=1).dropna()
covid_enrich_df.head(3)
```

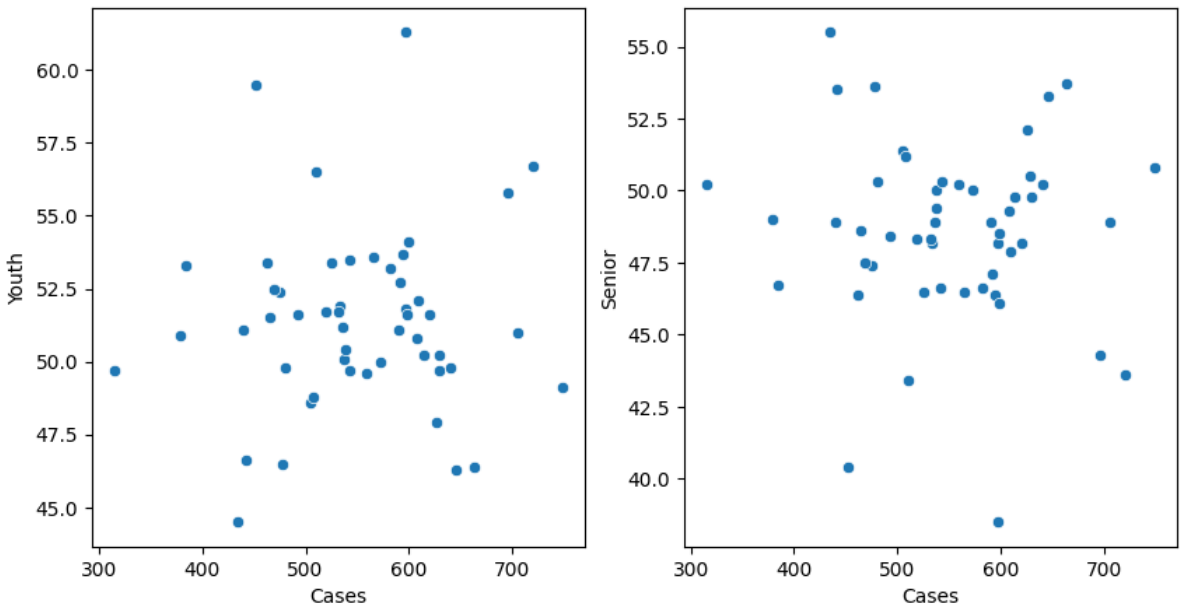
Out[ ]:

	Cases	Youth	Senior
State			
AK	720.0	56.7	43.6
AL	614.0	50.2	49.8
AR	597.0	51.8	48.2

In [ ]:
```
import seaborn as sns

fig, ax = plt.subplots(1, 2, figsize=(10, 5))
sns.scatterplot(x=covid_enrich_df.Cases, y=covid_enrich_df.Youth, ax=ax[0])
sns.scatterplot(x=covid_enrich_df.Cases, y=covid_enrich_df.Senior, ax=ax[1])
```

Out[ ]:
```
<Axes: xlabel='Cases', ylabel='Senior'>
```



In [ ]:
```
states_mean = covid_enrich_df.Cases.mean().round()
covid_youth_dom = covid_enrich_df[covid_enrich_df['Youth'] > 50].drop(['Youth', 'Se
```

Hypothesis Testing

Hypothesis #1:

- Hypothesis:
  - **Null Hypothesis H0** - Youth dominated states have similar new cases rate as the entire country
  - **Alternative Hypothesis H1** - Youth dominated states have different new cases rate from the entire country

In [ ]:
```
from scipy.stats import pearsonr
import scipy

covid_enrich_df['Youth'] = (covid_enrich_df['Youth']/50).astype(int)

corr, _ = pearsonr(covid_enrich_df.Cases.values, covid_enrich_df.Youth.values)
n = len(covid_enrich_df)
degrees_freedom = n - 1
r = corr

#calculate t-value
t = r * ((n - 2) / (1 - r**2))**0.5

#two tailed p-value
p = scipy.stats.t.sf(abs(t), degrees_freedom) * 2

print("Two-tailed t-test")
print("t-value:", t)
print("p-value:", p)

p1 = p/2
print("\nOne-tailed t-test")
print("p-value:", p1)
```

```
Two-tailed t-test
t-value: 0.15891566534075932
p-value: 0.8743754826126062

One-tailed t-test
p-value: 0.4371877413063031
```

- Results shows
  - The test statistic "t" is equal to 0.158
  - The PValue is 0.874 for the two tailed t-test
  - The PValue is 0.437 for the one tailed t-test
  - A high PValue shows that there is no significant difference between the population mean and the youth new cases mean

We should reject the Alternative Hypothesis H1