Project3

April 29, 2022

[1]:
```python
# Exercise 1a

import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt

df = pd.read_csv('08_gap-every-five-years.tsv', sep = '\t')
df.head()
```

[1]:
```
       country continent  year  lifeExp       pop   gdpPercap
0  Afghanistan      Asia  1952   28.801   8425333  779.445314
1  Afghanistan      Asia  1957   30.332   9240934  820.853030
2  Afghanistan      Asia  1962   31.997  10267083  853.100710
3  Afghanistan      Asia  1967   34.020  11537966  836.197138
4  Afghanistan      Asia  1972   36.088  13079460  739.981106
```

[2]:
```python
# Exercise 1b

life_exp_per_year = df.lifeExp
years = df.year

fig, ax = plt.subplots()
table = sb.violinplot(y=life_exp_per_year, x=years, data = df)

table.set_xlabel("Year")
table.set_ylabel("Life Expectancy")
table.set_title("Violin Plot Example")
fig.savefig("violin.png")
```

Question 1

According to the graph above, it seems that there exists a general trend of increasing life expentancy over time, and the trend is likely linear. This can be observed from the mean and maximum life expectancy from 1952 to 2007.

Question 2

The distribution of life expectancy across countries does not seem to be skewed to the left. It does not seem to be symmetric around its center nor is unimodal.

Question 3

I can observe, from the plot graph, that the mean value of life expectancy across countries steadily increase over time. Although there may be some countries that have decreasing or straight lines on the average life expectancy over time, because of this strong general trend, I think the chance of null hypothesis of no relationship between life expectancy and time (years) being true would likely be less than 5%. Thus, I would reject the null hypothesis rather than accepting it.

Question 4

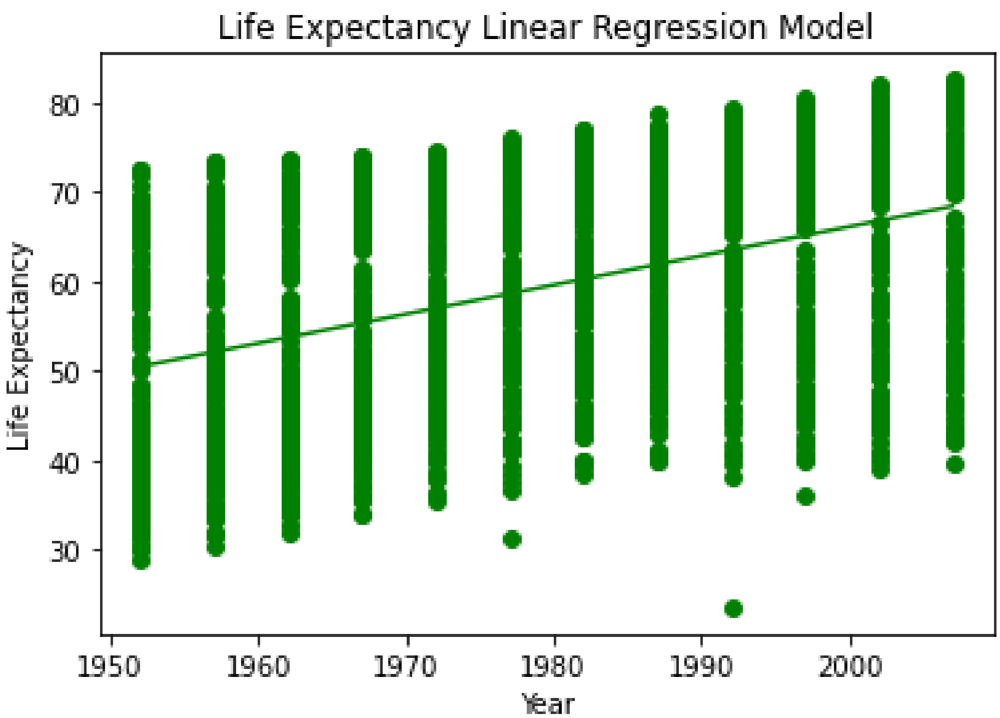
Except for some significant number of countries that seem to have the declined average life expectancy in years around 1977 and 1992, the residuals would appear almost as a linear function.

Question 5

I think the assumptions of the linear regression model strongly depends on the purpose of analyzing the data. Since my purpose is to show the average life expectancy across countries over the past

50 years, I do think that the assumptions such as noise and input distributions are appropriate.

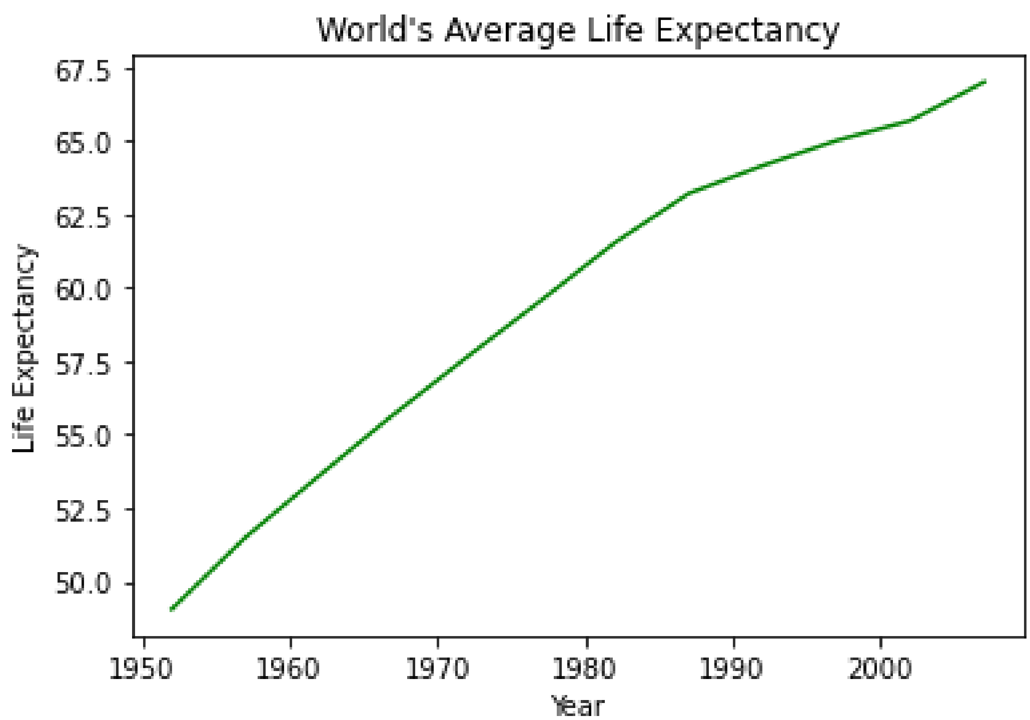
[3]:
```
# Exercise 2

import numpy as np
from sklearn.linear_model import LinearRegression

lrm = LinearRegression()
x = [[x] for x in df['year'].values]
y = [[y] for y in df['lifeExp'].values]
lrm_fit = lrm.fit(x, y)
y_predicted = lrm.predict(x)

plt.scatter(x, y, color = "green")
plt.plot(x, y_predicted, color = "green")
plt.xlabel("Year")
plt.ylabel("Life Expectancy")
plt.title("Life Expectancy Linear Regression Model")
plt.show()
```



[4]:
```
# Question 6a

years = df["year"].drop_duplicates()
average_per_year = []
```

```python
for year in years:
    life_exp = df.loc[df["year"] == year, "lifeExp"]
    average = 0
    for i in life_exp:
        average += i
    average /= len(life_exp)
    average_per_year.append(average)

plt.plot(years, average_per_year, color = "green")
plt.xlabel("Year")
plt.ylabel("Life Expectancy")
plt.title("World's Average Life Expectancy")
```

[4]: Text(0.5, 1.0, "World's Average Life Expectancy")



Question 6b

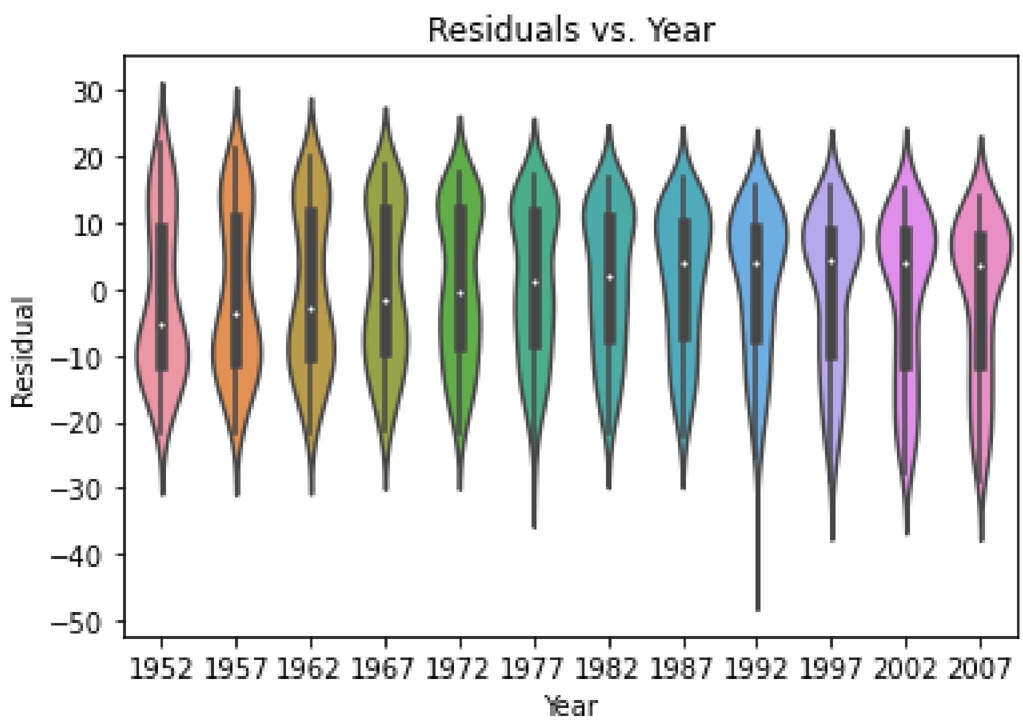
On average, world's average life expectancy would increase by 1.49 years ((2.45+2.1+2.07+1.97+1.93+1.96+1.68+0.95+0.85+0.68+1.31)/12) years.

Question 7

Yes, I would still reject the null hypothesis of no relationship between year and life expectancy. According to the world's average life expectancy over time, the average life expectancy seems to increase constantly as years go by.

[5]:
```
# Exercise 3

residuals = []
years = []

for i in range(0, len(y_predicted)):
    residuals.append(y[i][0] - y_predicted[i][0])

for j in x:
    years.append(j[0])

table3 = sb.violinplot(x = years, y = residuals, data = df)
table3.set_xlabel("Year")
table3.set_ylabel("Residual")
table3.set_title("Residuals vs. Year")
```

[5]: Text(0.5, 1.0, 'Residuals vs. Year')

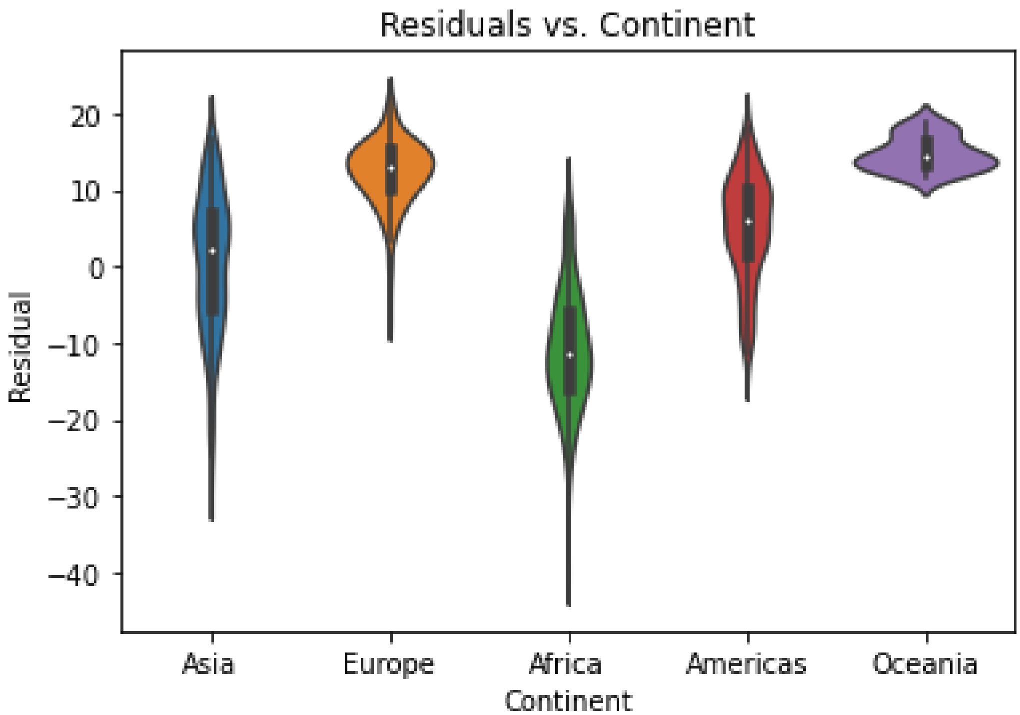


Question 8

Yes, the graph above shows a linear function as I expected it to be.

[6]:
```python
# Exercise 4

table4 = sb.violinplot(x = 'continent', y = residuals, data = df)
table4.set_xlabel("Continent")
table4.set_ylabel("Residual")
table4.set_title("Residuals vs. Continent")
```

[6]: Text(0.5, 1.0, 'Residuals vs. Continent')

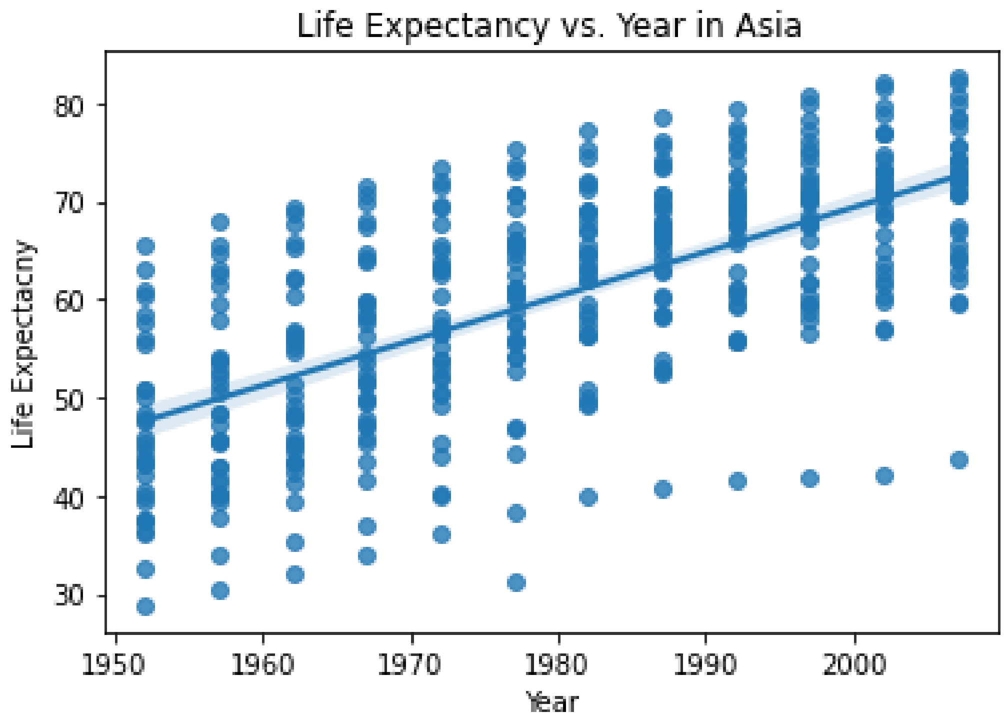


Question 9

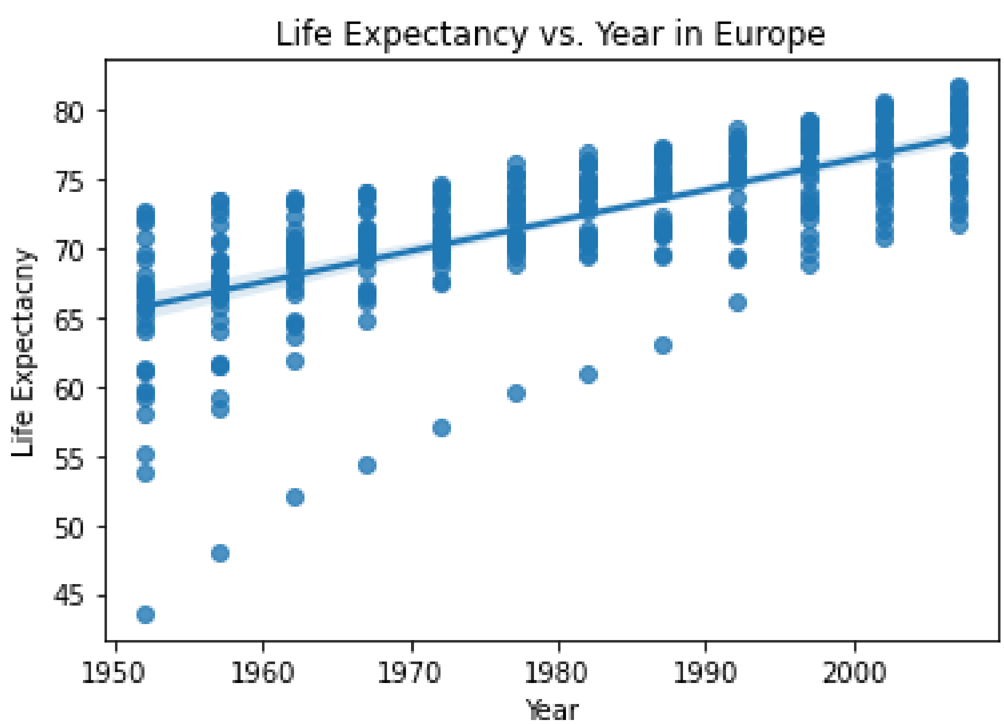
Yes, depending on which continent we are observing at, the average life expectancy seems to differ in accordance to its continent.

[7]:
```python
# Exercise 5

continents = df["continent"].drop_duplicates()

for conti in continents:
    new_table = df.loc[df["continent"] == conti]
    table5 = sb.regplot(x = "year", y = "lifeExp", data = new_table)
    table5.set_xlabel("Year")
    table5.set_ylabel("Life Expectacny")
    table5.set_title("Life Expectancy vs. Year in " + conti)
```

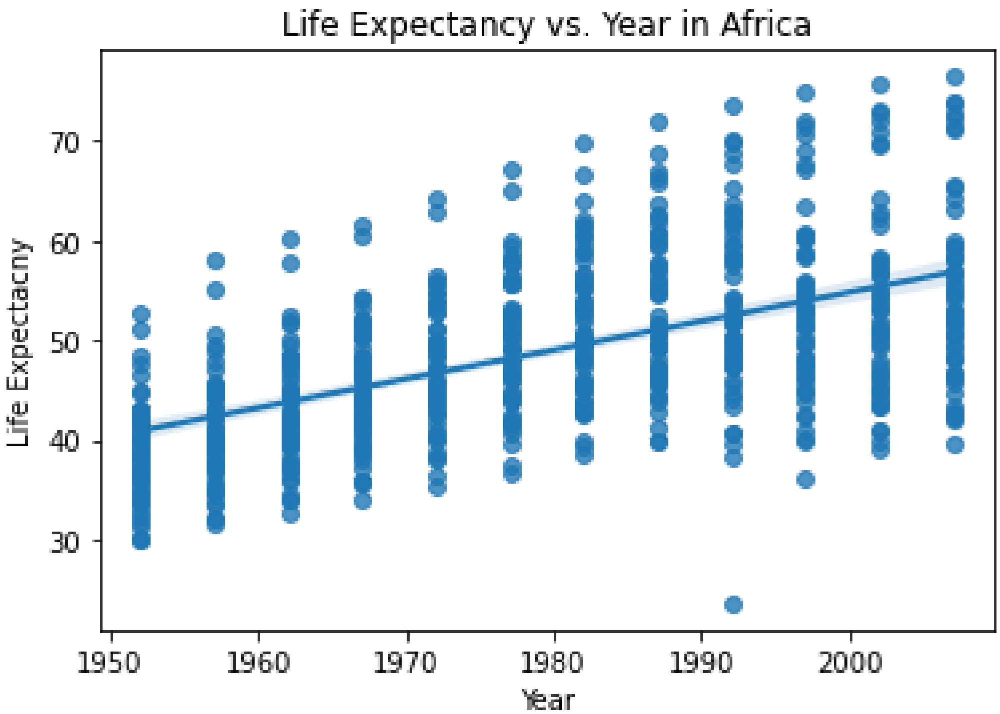
```
    plt.show()
```

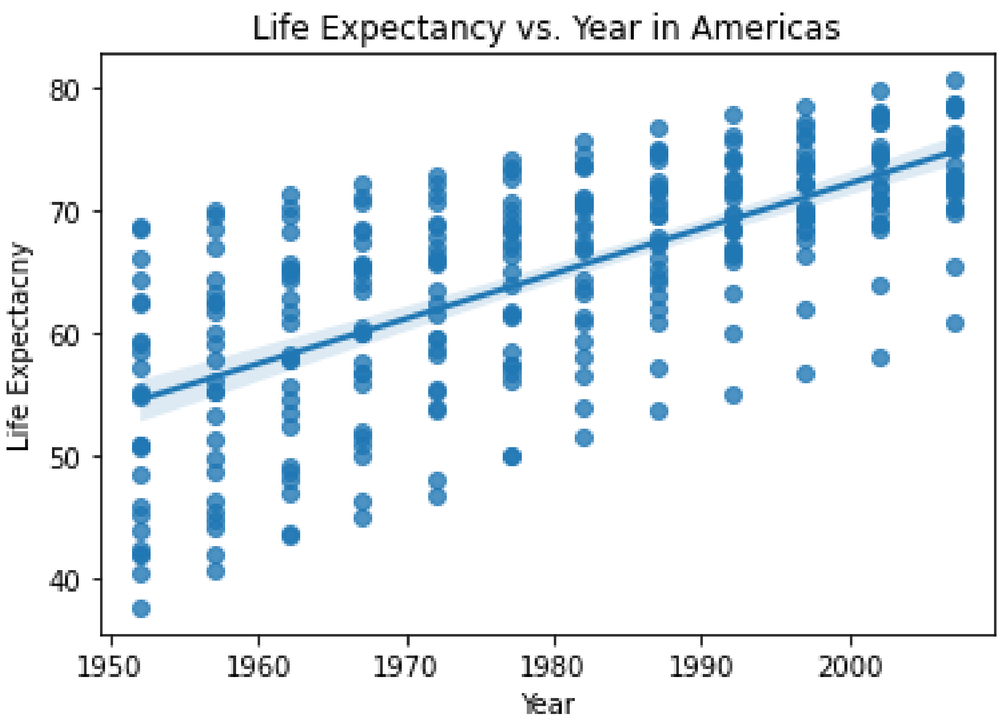


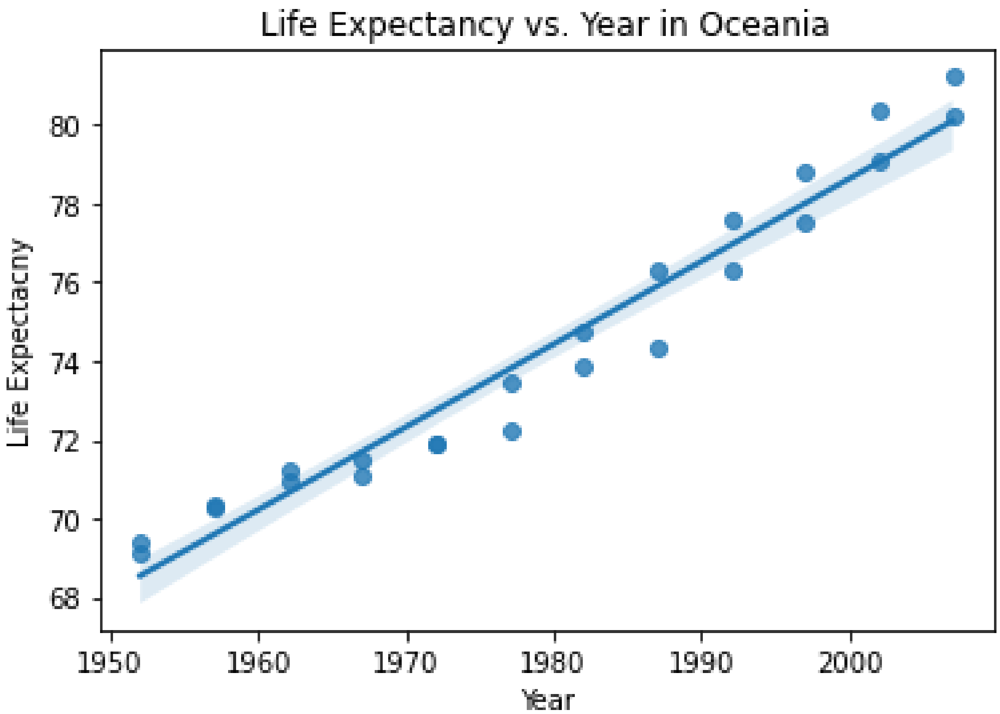
Life Expectancy vs. Year in Asia
Life Expectacny
Year
80
70
60
50
40
30
1950
1960
1970
1980
1990
2000



Life Expectancy vs. Year in Europe
Life Expectacny
Year
80
75
70
65
60
55
50
45
1950
1960
1970
1980
1990
2000


Life Expectancy vs. Year in Africa
Life Expectacny
Year
30
40
50
60
70
1950
1960
1970
1980
1990
2000


Life Expectancy vs. Year in Americas
Life Expectacny
Year
40
50
60
70
80
1950
1960
1970
1980
1990
2000



Question 10

Yes, the regression model shows that life expectancy is strongly dependent on years and continents as well. Thus, there should exist such an interaction term.

[8]:
```
# Exercise 6

import statsmodels.formula.api as smf

print(smf.ols(formula = 'lifeExp ~ year * continent', data = df).fit().
  ↪summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                lifeExp   R-squared:                       0.693
Model:                            OLS   Adj. R-squared:                  0.691
Method:                 Least Squares   F-statistic:                     424.3
Date:                Fri, 29 Apr 2022   Prob (F-statistic):               0.00
Time:                        02:20:37   Log-Likelihood:                -5771.9
No. Observations:                1704   AIC:                         1.156e+04
Df Residuals:                    1694   BIC:                         1.162e+04
Df Model:                           9
```

```
Covariance Type:            nonrobust
================================================================================
==============
                                 coef    std err          t      P>|t|
[0.025      0.975]
--------------------------------------------------------------------------------
--------------
Intercept                   -524.2578     32.963    -15.904      0.000
-588.911    -459.605
continent[T.Americas]       -138.8484     57.851     -2.400      0.016
-252.315     -25.382
continent[T.Asia]           -312.6330     52.904     -5.909      0.000
-416.396    -208.870
continent[T.Europe]          156.8469     54.498      2.878      0.004
49.957     263.737
continent[T.Oceania]         182.3499    171.283      1.065      0.287
-153.599     518.298
year                           0.2895      0.017     17.387      0.000
0.257       0.322
year:continent[T.Americas]     0.0781      0.029      2.673      0.008
0.021       0.135
year:continent[T.Asia]         0.1636      0.027      6.121      0.000
0.111       0.216
year:continent[T.Europe]      -0.0676      0.028     -2.455      0.014
-0.122      -0.014
year:continent[T.Oceania]     -0.0793      0.087     -0.916      0.360
-0.249       0.090
==============================================================================
Omnibus:                       27.121   Durbin-Watson:                   0.242
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               44.106
Skew:                          -0.121   Prob(JB):                     2.65e-10
Kurtosis:                       3.750   Cond. No.                     2.09e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 2.09e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
```

```
/opt/conda/lib/python3.9/site-packages/statsmodels/compat/pandas.py:65:
FutureWarning: pandas.Int64Index is deprecated and will be removed from pandas
in a future version. Use pandas.Index with the appropriate dtype instead.
  from pandas import Int64Index as NumericIndex
```

Question 11

All parameters in the model do have significantly different p-values from zero except for the one for Oceania. The p-value differ by 0.36 from zero for Oceania, and thus, all parameters except for one are significantly different from zero.

[9]:
```
# Question 12a

print(smf.ols(formula = 'lifeExp ~ year * continent', data = df).fit().params)
```

```
Intercept                      -524.257846
continent[T.Americas]          -138.848447
continent[T.Asia]              -312.633049
continent[T.Europe]             156.846852
continent[T.Oceania]            182.349883
year                              0.289529
year:continent[T.Americas]        0.078122
year:continent[T.Asia]            0.163593
year:continent[T.Europe]         -0.067597
year:continent[T.Oceania]        -0.079257
dtype: float64
```

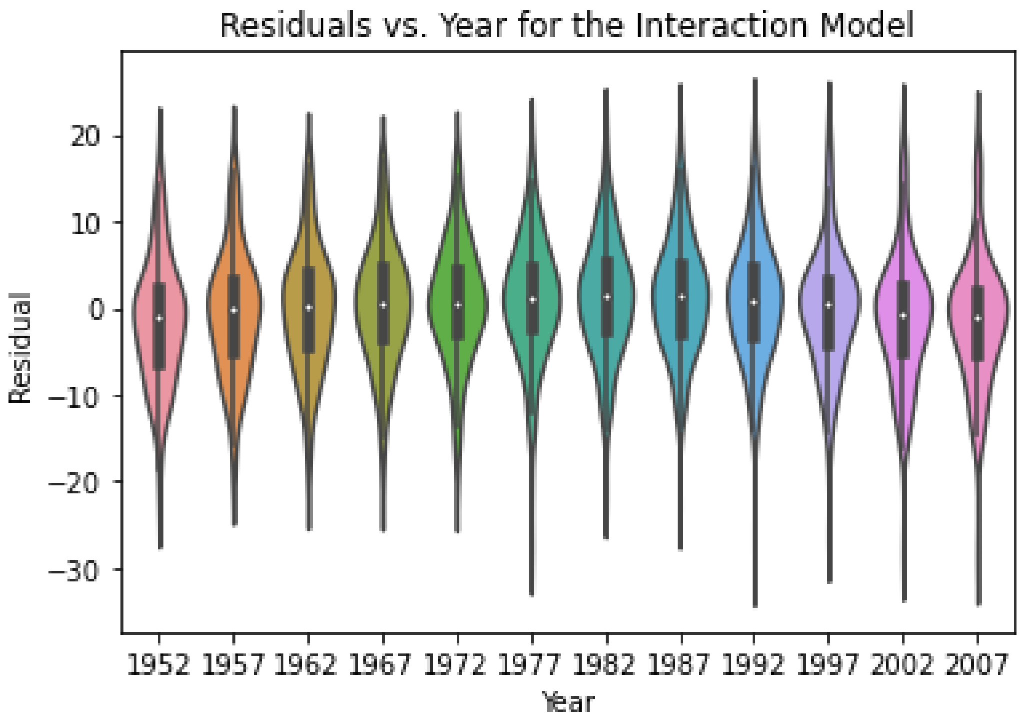
Question 12b

On average, life expectancy increased by 0.29 each year in Africa, 0.078 in America; 0.164 in Asia, -0.067 in Europe, and -0.079 in Oceania.

[10]:
```
# Exercise 7

reg_model = smf.ols(formula = 'lifeExp ~ year * continent', data = df).fit()

df["interaction"] = df.loc[df["continent"] == "Africa", "lifeExp"] - (reg_model.
 ↪params[0] + reg_model.params[5] * df.loc[df["continent"] == "Africa",␣
 ↪"year"])
df.loc[df["continent"] == "Americas", "interaction"] = df.loc[df["continent"]␣
 ↪== "Americas", "lifeExp"] - (reg_model.params[0] + reg_model.params[1] +␣
 ↪(reg_model.params[5] + reg_model.params[6]) * df.loc[df["continent"] ==␣
 ↪"Americas", "year"])
df.loc[df["continent"] == "Asia", "interaction"] = df.loc[df["continent"] ==␣
 ↪"Asia", "lifeExp"] - (reg_model.params[0] + reg_model.params[2] + (reg_model.
 ↪params[5] + reg_model.params[7]) * df.loc[df["continent"] == "Asia", "year"])
df.loc[df["continent"] == "Europe", "interaction"] = df.loc[df["continent"] ==␣
 ↪"Europe", "lifeExp"] - (reg_model.params[0] + reg_model.params[3] +␣
 ↪(reg_model.params[5] + reg_model.params[8]) * df.loc[df["continent"] ==␣
 ↪"Europe", "year"])
df.loc[df["continent"] == "Oceania", "interaction"] = df.loc[df["continent"] ==␣
 ↪"Oceania", "lifeExp"] - (reg_model.params[0] + reg_model.params[4] +␣
 ↪(reg_model.params[5] + reg_model.params[9]) * df.loc[df["continent"] ==␣
 ↪"Oceania", "year"])

table7 = sb.violinplot(x = "year", y = "interaction", data = df)
table7.set_xlabel("Year")
```

```
table7.set_ylabel("Residual")
table7.set_title("Residuals vs. Year for the Interaction Model")

# The result graph for the interaction model seems to match with the linear
  ↪regression models life expectancy vs. year
# and life expectancy vs. continent.
```

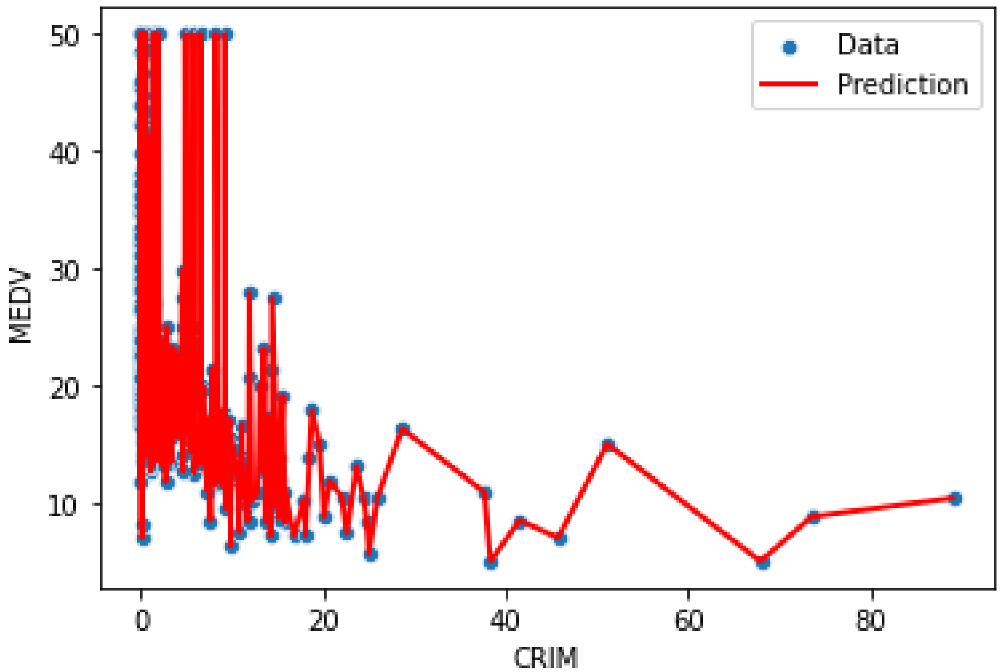
[10]: Text(0.5, 1.0, 'Residuals vs. Year for the Interaction Model')



[14]:
```
## Part 2: Decision Tree
# Crime Rate vs. Housing Price

import seaborn as sns
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.metrics import make_scorer, r2_score
from sklearn.model_selection import train_test_split
from numpy import mean
from numpy import std

# Extract and create a table from the data file
data = pd.read_csv('housing.data', delimiter=r"\s+")
```

```python
# Extract and set the data trains
X = data[["CRIM"]]
y = data["MEDV"]

# Create a decision tree regression model
dtr_model = DecisionTreeRegressor(max_depth = 30, random_state = 1)
dtr_model.fit(X, y)

# Show the graph for the model
sns.scatterplot(x = data["CRIM"], y = data["MEDV"], label = "Data")
plt.plot(data["CRIM"].sort_values(), dtr_model.predict(data["CRIM"].
 ↪sort_values().to_frame()), color = "red", label = "Prediction", linewidth =␣
 ↪2)
plt.legend()

# Use a 10-fold cross-validation
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30,␣
 ↪random_state = 42, shuffle = True)
score = make_scorer(r2_score)
ten_fold = KFold(n_splits = 10, random_state = 1, shuffle = True)
scores = cross_val_score(dtr_model, X, y, cv = ten_fold, scoring = score)
print("Score Mean: ", scores.mean(), "  Score Standard Deviation", scores.std())

# Show accuracy
print("Accuracy: ", dtr_model.score(X_train, y_train))
```

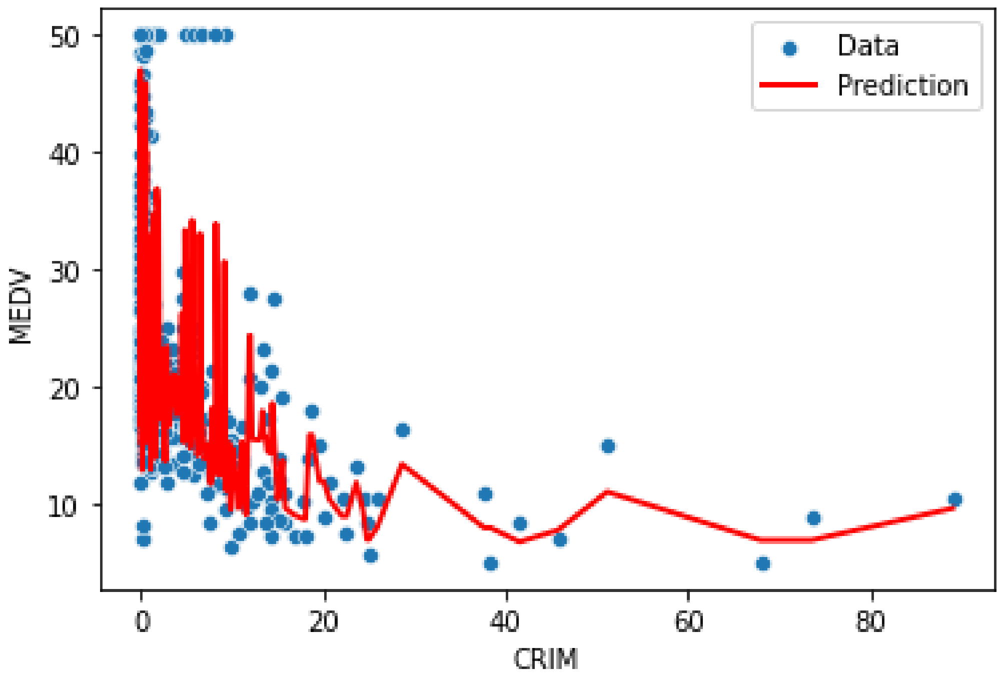
```
Score Mean:  -0.4671651256225447   Score Standard Deviation 0.3789322670917842
Accuracy:  0.9885431224562662
```



[15]:
```
## Part 2: k-NN
# Crime Rate vs. Housing Price
from sklearn.neighbors import KNeighborsRegressor

# Extract and create a table from the data file
data = pd.read_csv('housing.data', delimiter=r"\s+")

# Extract and set the data trains
X = data[["CRIM"]]
y = data["MEDV"]

# Create a k-NN model
knn_model = KNeighborsRegressor(n_neighbors = 2)
knn_model.fit(X, y)

# Show the graph for the model
sns.scatterplot(x = data["CRIM"], y = data["MEDV"], label = "Data")
plt.plot(data["CRIM"].sort_values(), knn_model.predict(data["CRIM"].
 ↪sort_values().to_frame()), color = "red", label = "Prediction", linewidth =␣
 ↪2)
plt.legend()

# Use a 10-fold cross-validation
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30,␣
 ↪random_state = 42, shuffle = True)

score = make_scorer(r2_score)
ten_fold = KFold(n_splits = 10, random_state = 1, shuffle = True)
scores = (cross_val_score(knn_model, X, y, cv = ten_fold, scoring = score))
print("Score Mean: ", scores.mean(), "  Score Standard Deviation", scores.std())

# Show accuracy
print("Accuracy: ", knn_model.score(X_train, y_train))
```

```
Score Mean:  -0.01340069039535191   Score Standard Deviation 0.22086652684964203
Accuracy:  0.6566782103159237
```



I am using the housing data to experiment the relationship between CRIM (crime rate) and MEDV (housing price). I classified the data using the two algorithms which are Decision Tree Regression and k_NN Regression. I found out that predicting the housing price in accordance to crime rates in towns seems appropriate after seeing the results. Predicting the housing price seems to be more effective with the Decision Tree algorithm because it shows greater accuracy than the k_NN algorithm. I think it is because the size of the data is too large for the k_NN algorithm that it makes the prediction weak. With the codes above, I showed that crime rates do affect housing prices in town, and that predicting it with the Decision Tree Regression model is more effective.