Project3

December 4, 2020

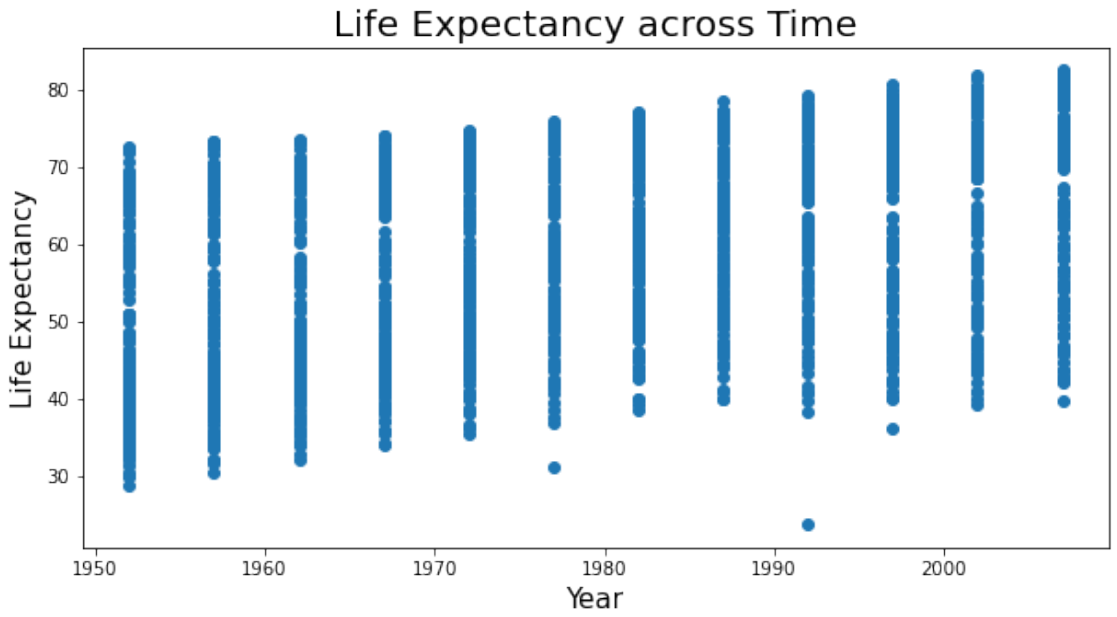
[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

data = pd.read_csv("gap.tsv", sep='\t')
data.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

Part 1 Exercise 1: Make a scatter plot of life expectancy across time.

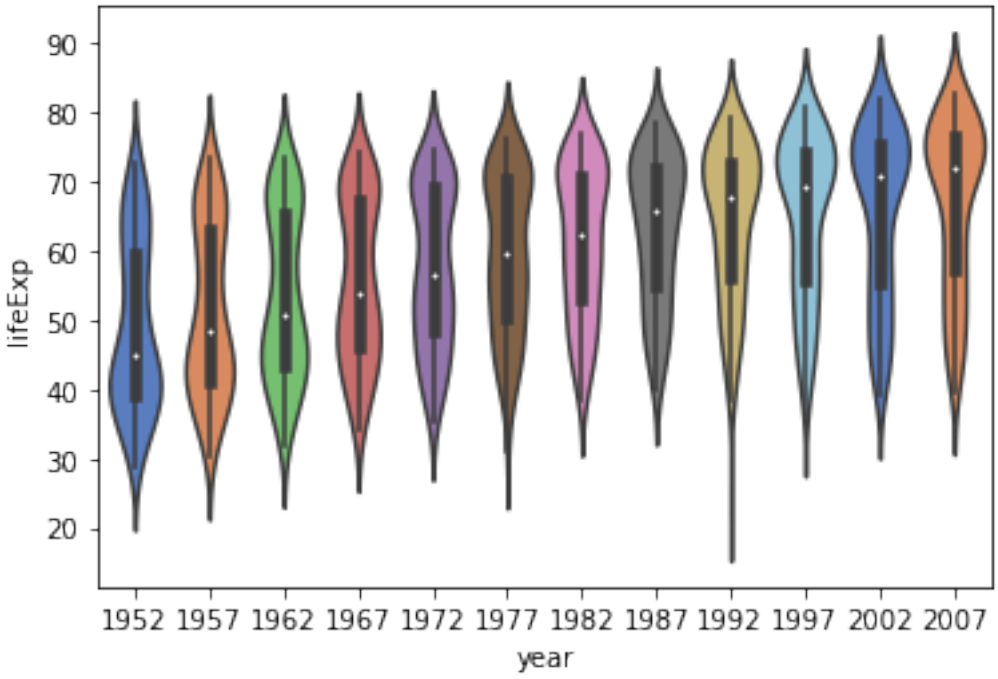
[2]:
```
plt.figure(figsize=(10,5))
plt.scatter(data['year'], data['lifeExp'])
plt.xlabel("Year", fontsize=15)
plt.ylabel("Life Expectancy", fontsize=15)
plt.title("Life Expectancy across Time", fontsize=20)

plt.show()
```



Question 1: Is there a general trend (e.g., increasing or decreasing) for life expectancy across time? Is this trend linear? (answering this qualitatively from the plot, you will do a statistical analysis of this question shortly)

[3]:
```
# Generally speaking, the life expectancy is increasing across time. The trend␣
↪is linear.
```

[4]:
```
import seaborn as sns
ax = sns.violinplot(x="year", y="lifeExp",
                    data=data, palette="muted", split=True)
```



Question 2: How would you describe the distribution of life expectancy across countries for individual years? Is it skewed, or not? Unimodal or not? Symmetric around it's center?

[5]:
```
# The life expectancy is increasing across time. Before 1977, the plot is␣
↪bottom-heavy. After that, the plots become more
# top-heavy.
# It is not sysmetric. After year 1977 the distribution becomes more skewed so␣
↪we can see tails.
# Year 1957 to 1977 looks bimodal, and the rest of the distribution looks␣
↪unimodal.
```

Question 3: Suppose I fit a linear regression model of life expectancy vs. year (treating it as a continuous variable), and test for a relationship between year and life expectancy, will you reject the null hypothesis of no relationship? (do this without fitting the model yet. I am testing your intuition.)

[6]:
```
# Yes. Based on the violin plot, we can see the life expectancy increases over␣
↪time. So there is a relationship between them.
```

Question 4: What would a violin plot of residuals from the linear model in Question 3 vs. year look like? (Again, don't do the analysis yet, answer this intuitively)

[7]:
```
# Probably similar to the plot in exercise 1.
```

Question 5: According to the assumptions of the linear regression model, what should that violin plot look like? That is, consider the assumptions the linear regression model you used assumes

(e.g., about noise, about input distributions, etc); do you think everything is okay?

[8]:
```
# May have an increasing trend.
```

Exercise 2: Fit a linear regression model using, e.g., the LinearRegression function from Scikit-Learn or the closed-form solution we derived in class, for life expectancy vs. year (as a continuous variable). There is no need to plot anything here, but please print the fitted model out in a readable format.

[9]:
```
from sklearn.linear_model import LinearRegression
reg = LinearRegression().fit(data[["year"]], data["lifeExp"])
```

Question 6: On average, by how much does life expectancy increase every year around the world?

[10]:
```
print(reg.coef_[0])
print("life expectancy increase every year around the world by 0.
↪3259038276371518")
```

```
0.3259038276371518
life expectancy increase every year around the world by 0.3259038276371518
```

Question 7: Do you reject the null hypothesis of no relationship between year and life expectancy? Why?

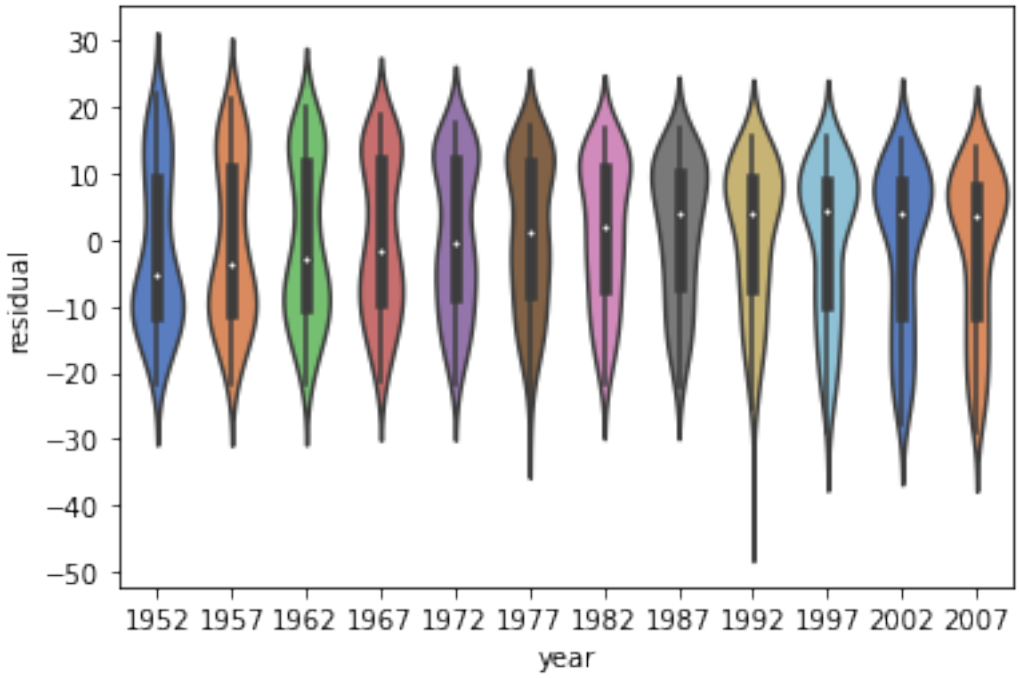
[11]:
```
import statsmodels.api as sm
import statsmodels.formula.api as smf
results = smf.ols('lifeExp  ~ year', data=data).fit()
print(results.summary())
# Yes, p-value is small enough.
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                lifeExp   R-squared:                       0.190
Model:                            OLS   Adj. R-squared:                  0.189
Method:                 Least Squares   F-statistic:                     398.6
Date:                Fri, 04 Dec 2020   Prob (F-statistic):           7.55e-80
Time:                        03:13:07   Log-Likelihood:                -6597.9
No. Observations:                1704   AIC:                         1.320e+04
Df Residuals:                    1702   BIC:                         1.321e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept   -585.6522     32.314    -18.124      0.000    -649.031    -522.273
year           0.3259      0.016     19.965      0.000       0.294       0.358
==============================================================================
Omnibus:                      386.124   Durbin-Watson:                   0.197
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               90.750
```

```
Skew:                          -0.268   Prob(JB):                     1.97e-20
Kurtosis:                       2.004   Cond. No.                     2.27e+05
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 2.27e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Exercise 3: Make a violin plot of residuals vs. year for the linear model from Exercise 2.

[12]:
```
m = reg.coef_[0]
h = reg.intercept_
data["residual"] =  data["lifeExp"] - (data["year"] * m + h)

ax2 = sns.violinplot(x="year", y="residual",
                     data=data, palette="muted", split=True)
```

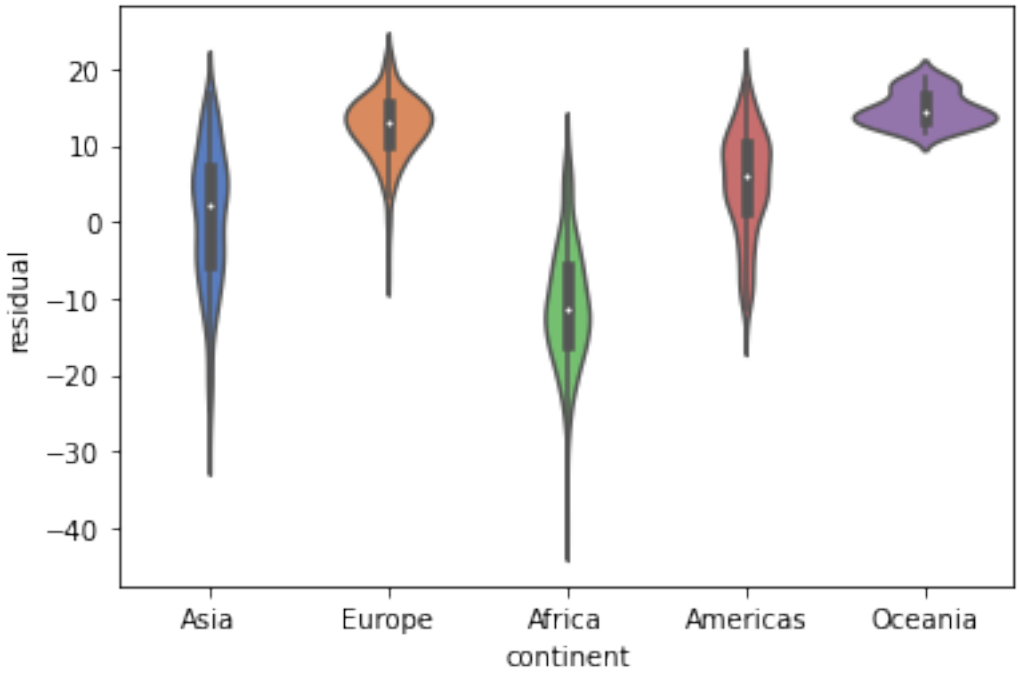


Question 8: Does the plot of Exercise 3 match your expectations (as you answered Question 4)?

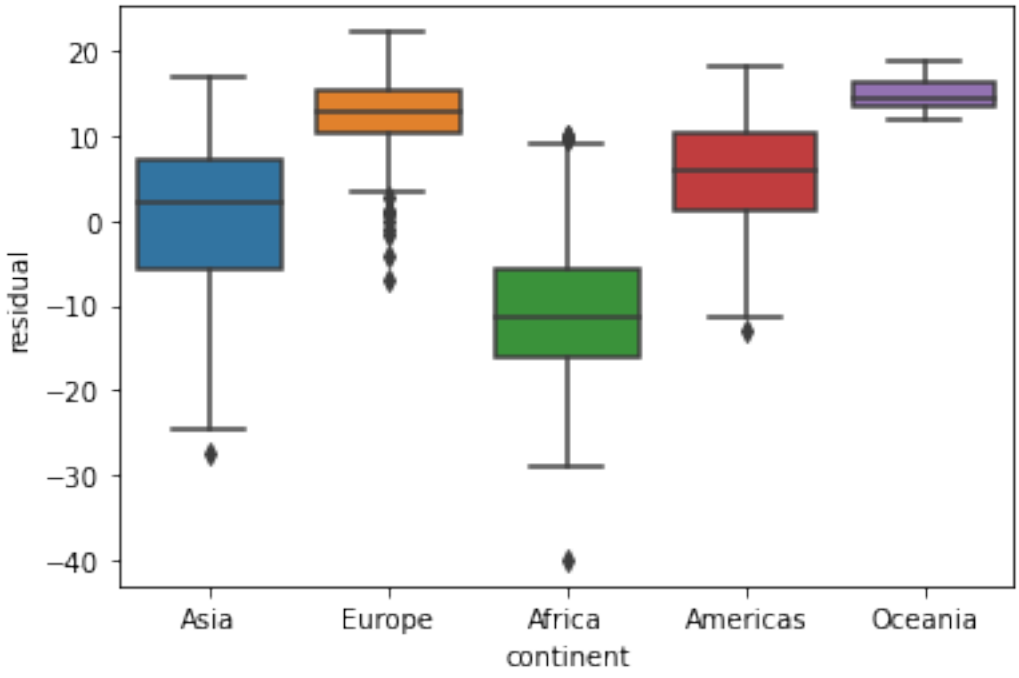
[13]:
```
# No, it has a trend of decreasing.
```

Exercise 4: Make a boxplot (or violin plot) of model residuals vs. continent.

[14]:
```
ax_violin = sns.violinplot(x="continent", y="residual",
                           data=data, palette="muted", split=True)
```



[15]:
```
ax_box = sns.boxplot(x="continent", y="residual", data=data)
```



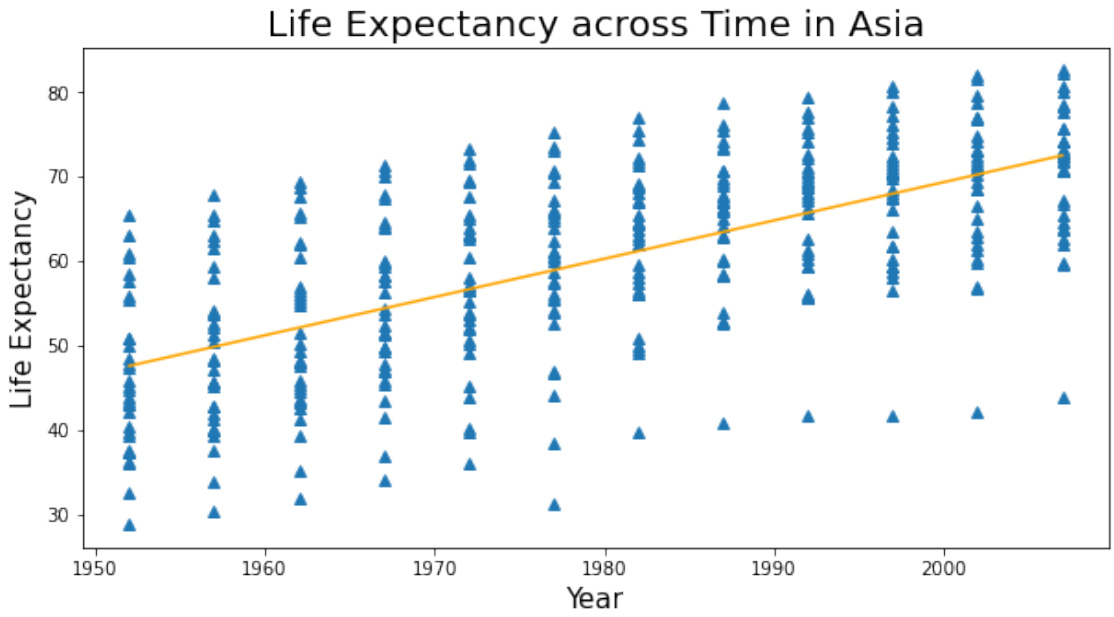
Question 9: Is there a dependence between model residual and continent? If so, what would that suggest when performing a regression analysis of life expectancy across time?

[16]:
```
# No, I can't see any relationship between continent and residual from the plot.
```

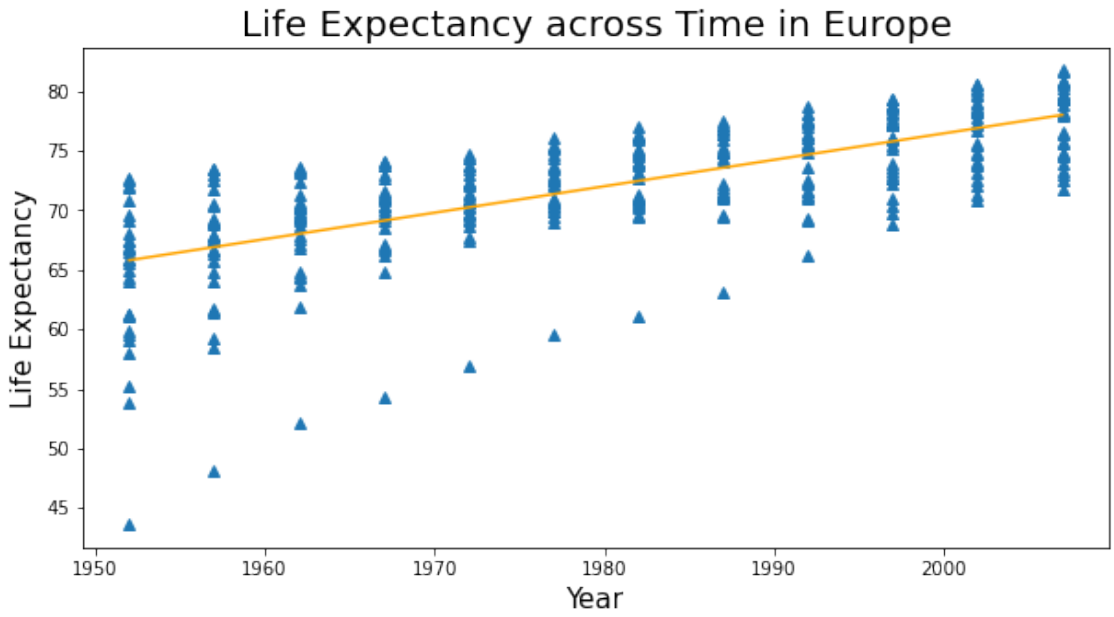
Exercise 5: As in the Moneyball project, make a scatter plot of life expectancy vs. year, grouped by continent, and add a regression line. The result here can be given as either one scatter plot per continent, each with its own regression line, or a single plot with each continent's points plotted in a different color, and one regression line per continent's points. The former is probably easier to code up.

[17]:
```
continents = data.groupby('continent')
```

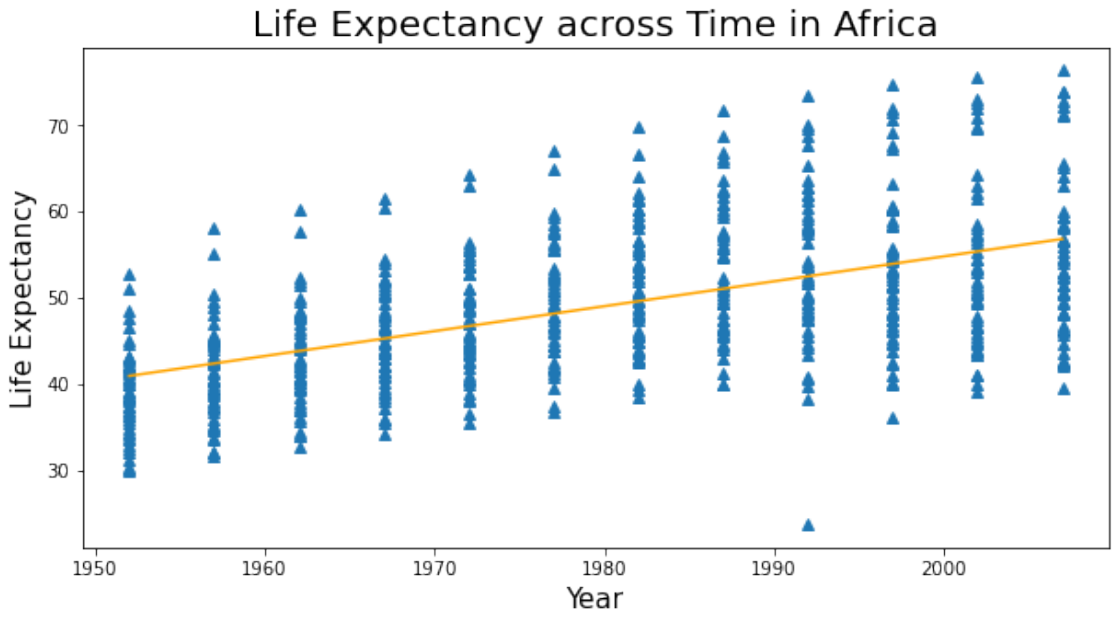
[18]:
```
# Asia
asia = continents.get_group('Asia')
plt.figure(figsize=(10,5))
plt.plot(asia['year'], asia['lifeExp'], "^")
plt.xlabel("Year", fontsize=15)
plt.ylabel("Life Expectancy", fontsize=15)
plt.title("Life Expectancy across Time in Asia", fontsize=20)

x_asia = [[x] for x in asia['year'].values]
y_asia = [[y] for y in asia['lifeExp'].values]
linearRegressor = LinearRegression()
reg_asia = linearRegressor.fit(x_asia, y_asia)

plt.plot(x_asia, linearRegressor.predict(x_asia), color='orange')

plt.show()
```



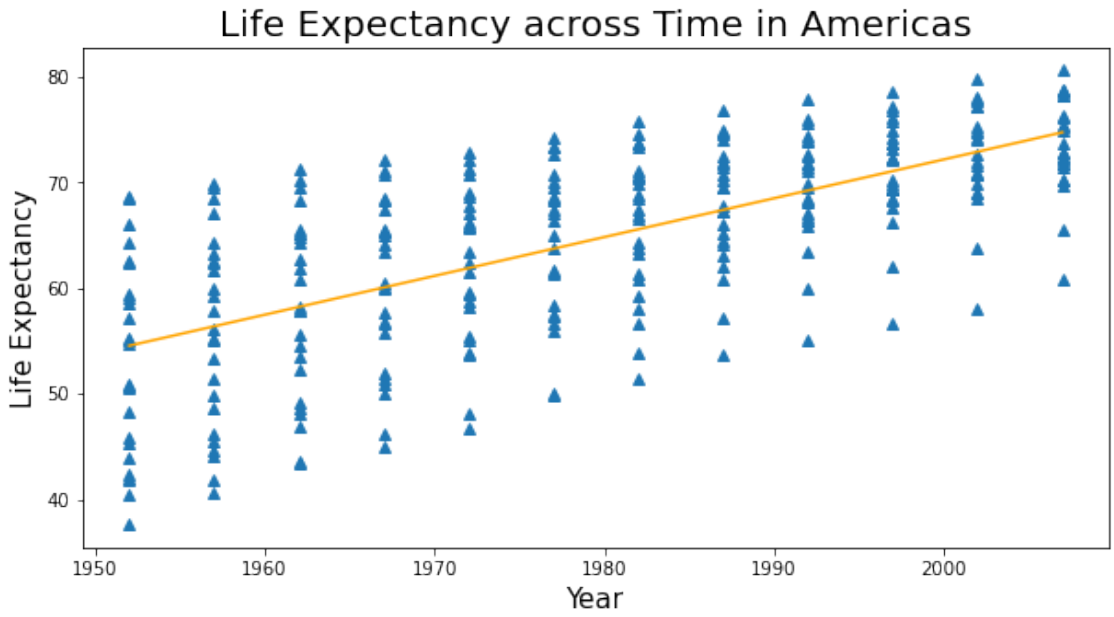
[19]:
```
# Europe
europe = continents.get_group('Europe')
plt.figure(figsize=(10,5))
plt.plot(europe['year'], europe['lifeExp'], "^")
plt.xlabel("Year", fontsize=15)
plt.ylabel("Life Expectancy", fontsize=15)
plt.title("Life Expectancy across Time in Europe", fontsize=20)

x_europe = [[x] for x in europe['year'].values]
y_europe = [[y] for y in europe['lifeExp'].values]
linearRegressor = LinearRegression()
reg_europe = linearRegressor.fit(x_europe, y_europe)

plt.plot(x_europe, linearRegressor.predict(x_europe), color='orange')

plt.show()
```



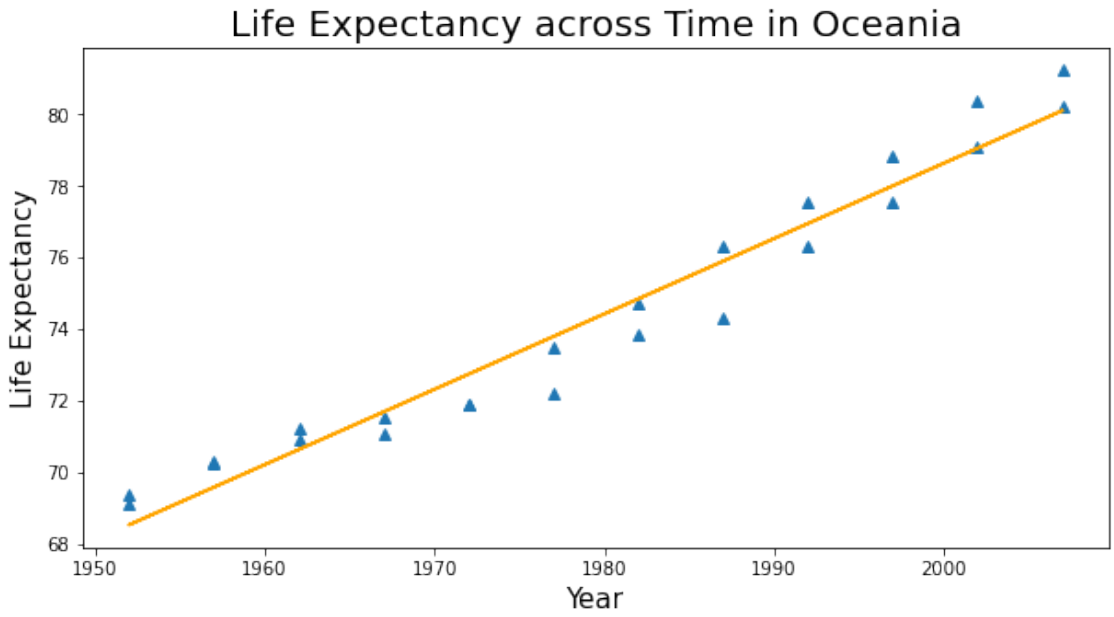
[20]:
```
# Africa
africa = continents.get_group('Africa')
plt.figure(figsize=(10,5))
plt.plot(africa['year'], africa['lifeExp'], "^")
plt.xlabel("Year", fontsize=15)
plt.ylabel("Life Expectancy", fontsize=15)
plt.title("Life Expectancy across Time in Africa", fontsize=20)

x_africa = [[x] for x in africa['year'].values]
y_africa = [[y] for y in africa['lifeExp'].values]
linearRegressor = LinearRegression()
reg_africa = linearRegressor.fit(x_africa, y_africa)

plt.plot(x_africa, linearRegressor.predict(x_africa), color='orange')

plt.show()
```



[21]:
```
# Americas
americas = continents.get_group('Americas')
plt.figure(figsize=(10,5))
plt.plot(americas['year'], americas['lifeExp'], "^")
plt.xlabel("Year", fontsize=15)
plt.ylabel("Life Expectancy", fontsize=15)
plt.title("Life Expectancy across Time in Americas", fontsize=20)

x_americas = [[x] for x in americas['year'].values]
y_americas = [[y] for y in americas['lifeExp'].values]
linearRegressor = LinearRegression()
reg_americas = linearRegressor.fit(x_americas, y_americas)

plt.plot(x_americas, linearRegressor.predict(x_americas), color='orange')

plt.show()
```



[22]:
```
# Oceania
oceania = continents.get_group('Oceania')
plt.figure(figsize=(10,5))
plt.plot(oceania['year'], oceania['lifeExp'], "^")
plt.xlabel("Year", fontsize=15)
plt.ylabel("Life Expectancy", fontsize=15)
plt.title("Life Expectancy across Time in Oceania", fontsize=20)

x_oceania = [[x] for x in oceania['year'].values]
y_oceania = [[y] for y in oceania['lifeExp'].values]
linearRegressor = LinearRegression()
reg_oceania = linearRegressor.fit(x_oceania, y_oceania)

plt.plot(x_oceania, linearRegressor.predict(x_oceania), color='orange')

plt.show()
```



Question 10: Based on this plot, should your regression model include an interaction term for continent and year? Why?

[23]:
```
# That depends. Generally, all the continents have the same trends of␣
 ↪increasing life expectancy over time. But they increase
# with different rate.
```

Exercise 6: Fit a linear regression model for life expectancy including a term for an interaction between continent and year. Print out the model in a readable format, e.g., print the coefficients of the model (no need to plot). Hint: adding interaction terms is a form of feature engineering, like we discussed in class (think about, e.g., using (a subset of) polynomial features here).

[24]:
```
results = smf.ols('lifeExp  ~ continent*year', data=data).fit()
print(results.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, 04 Dec 2020   Prob (F-statistic):               0.00
Time:                        03:13:08   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
continent[T.Americas]:year     0.0781      0.029      2.673      0.008
0.021       0.135
continent[T.Asia]:year         0.1636      0.027      6.121      0.000
0.111       0.216
continent[T.Europe]:year      -0.0676      0.028     -2.455      0.014
-0.122      -0.014
continent[T.Oceania]:year     -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
==============================================================================

Warnings:
[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.
```

Question 11: Are all parameters in the model significantly different from zero? If not, which are not significantly different from zero?

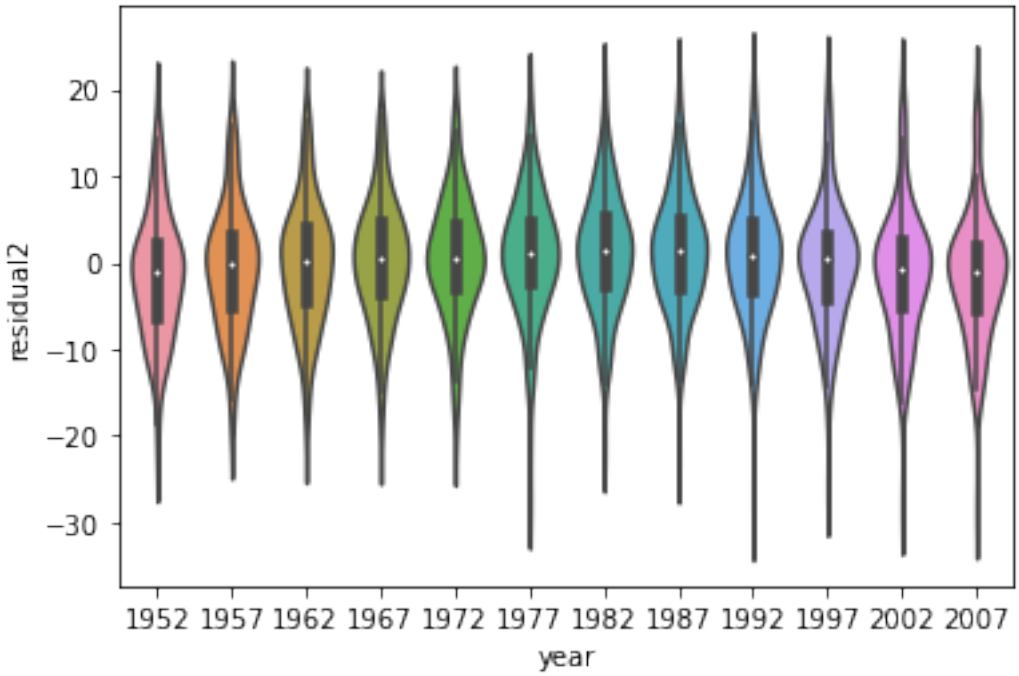
[25]:
```
# Only Oceania is not significantly different from zero since p is >= 0.05.
```

Question 12: On average, by how much does life expectancy increase each year for each continent? (Provide code to answer this question by extracting relevant estimates from model fit)

[26]:
```
# Americas 0.0781 + 0.2895 = 0.3676
# Asia: 0.1636 + 0.2895 = 0.4531
# Europe: -0.0676 + 0.2895 = 0.2219
# Oceania: -0.0793 + 0.2895 = 0.2102
# Africa: 0.2895
```

Exercise 8: Make a residuals vs. year violin plot for the interaction model. Comment on how well it matches assumptions of the linear regression model.

[27]:
```
data["residual2"] = data["lifeExp"] - results.fittedvalues
ax_res = sns.violinplot(x="year", y="residual2",
                        data=data)
```



0.0.1 Part 2

Problem 1 Implement the gradient descent algorithm (either batch or stochastic versions) for multiple linear regression. I.e., extend the version of the algorithm in the lecture notes to multiple parameters.

[28]:
```
# I think we can still use the same code provided in the slides
# Training data (X, y), T time steps, alpha step
def grad_descent(X, y, T, alpha):
    m, n = X.shape # m = #examples, n = #features
    theta = np.zeros(n) # initialize parameters
    f = np.zeros(T) # track loss over time
```

```
    for i in range(T):
        # loss for current parameter vector theta
        f[i] = 0.5*np.linalg.norm(X.dot(theta) - y)**2
        # compute steepest ascent at f(theta)
        g = X.T.dot(X.dot(theta) - y)
        # step down the gradient
        theta = theta - alpha*g
    return theta, f
```

Problem 2 (This problem is +5 points of extra credit.) Derive the above update equation. Write the derivation in a markdown ipynb cell.

[29]:
```
# See another file. Just plug in everything.
```

Problem 3 Implement the gradient descent algorithm (either batch or stochastic versions) for multiple logistic regression. I.e., modify your code in problem 1 for the logistic regression update equation. Make sure you include in your submission writeup, which version of the algorithm you are solving (stochastic or batch), and make sure to comment your code to help us understand your implementation.

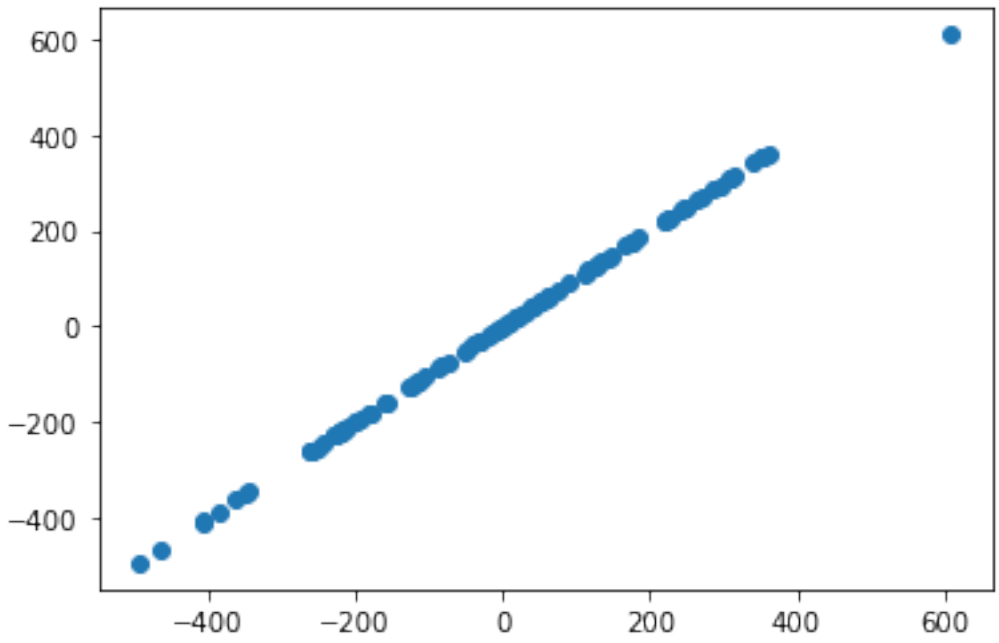
[30]:
```
# batch

# alpha: alpha
# theta: Beta
# y: y
# X: X

# Training data (X, y), T time steps, alpha step
def grad_descent_log(X, y, T, alpha):
    m, n = X.shape        # m = #examples, n = #features
    theta = np.zeros(n)   # initialize parameters
    g = 0

    for i in range(T):
        p = (np.exp(X.dot(theta)) / (1 + np.exp(X.dot(theta))))
        g = (y - p).dot(X)
        # step down the gradient
        theta = theta + alpha * g
    return theta
```

Problem 4 To test your programs, simulate data from the linear regression and logistic regression models and check that your implementations recover the simulation parameters properly.

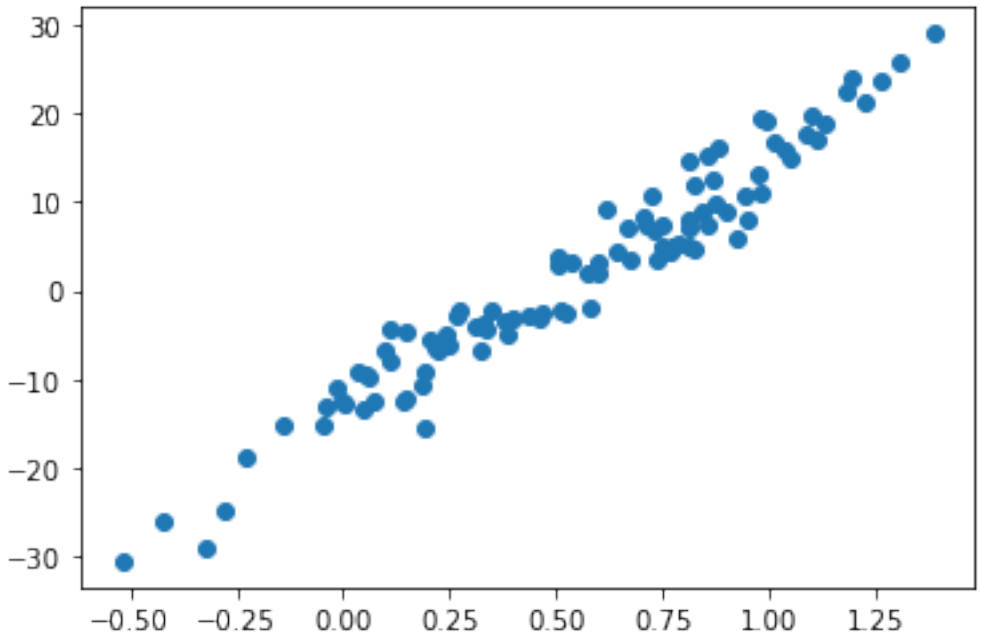
[31]:
```
from sklearn import linear_model
from sklearn import datasets
# Generate data for linear regression:
```

```
gen_data_x, gen_data_y = datasets.make_regression(n_samples=100, n_features=20,␣
 ↪noise = 1.5)

# Generate data for logistic regression. This is similar to linear, only now␣
 ↪values are either 0 or 1.
log_gen_data_x, dump_y = datasets.make_regression(n_samples=100, n_features=20,␣
 ↪noise = 1.5)
log_gen_data_y = [0 if i>0 else 1 for i in dump_y]

[theta, f] = grad_descent(gen_data_x, gen_data_y, 10000, .001)
log = grad_descent_log(log_gen_data_x, log_gen_data_y, 10000, .001)

linearRegressor = LinearRegression()

print(theta)
linearRegressor.fit(gen_data_x, gen_data_y)
predictions = linearRegressor.predict(gen_data_x)
plt.scatter(predictions, np.dot(gen_data_x, theta))
plt.show()

print(log)
linearRegressor.fit(log_gen_data_x, log_gen_data_y)
predictions_log = linearRegressor.predict(log_gen_data_x)
plt.scatter(predictions_log, np.dot(log_gen_data_x, log))
plt.show()
```

```
[-2.07553542e-01  1.13007993e+01  2.49419125e-01  9.40937395e+01
  2.46119738e-01 -2.21748955e-01  2.33542627e-01  9.74346134e+01
  1.11062419e-01  7.63459017e+01  1.26011722e-01  8.87641250e+01
  1.54757971e-01  8.21043520e+01  7.60530983e-02  4.61951023e+01
  3.09829139e+01  2.00412173e-02  2.07277295e+00  2.34172392e+01]
```




```
[-2.46145478e+00 -4.75329902e+00  3.69117035e-01  7.26944337e-01
 -6.83200433e-01 -1.05104933e+00 -4.27966541e+00 -2.65275932e+00
 -2.34605192e+00 -4.31654072e+00 -5.89712078e+00 -1.90130260e+00
 -6.96477232e-01 -2.67820843e-01 -1.08449076e+00 -3.50465056e+00
 -1.16184144e-01 -1.66984026e-01  5.98135209e-01  3.23588105e-03]
```



0.0.2 Try it out

The data is from https://corgis-edu.github.io/corgis/csv/cars/

I am trying to predict whether the width of a car influence horsepower using Linear Discriminant Analysis and linear SVM.

[32]:
```
car = pd.read_csv('cars.csv',delimiter=',')
car.columns = ["Height","Length","Width","Driveline","Engine Type","Is␣
↪Hybrid","Number of Forward Gears","Transmission",
               "City mpg","Fuel Type","Highway␣
↪mpg","Classification","ID","Brand","Model","Year","Horsepower","Torque"]
car = car.drop(['Brand','Year','Model','Classification','Fuel␣
↪Type','Transmission','Is Hybrid','Driveline','Engine Type'], axis = 1)
car = car[['ID', 'Height','Length','Width','Number of Forward Gears','City␣
↪mpg','Highway mpg','Horsepower','Torque']]
car = car.iloc[0:700, :]
car
```

[32]:

	ID	Height	Length	Width	\
0	2009 Audi A3 3.2	140	143	202	
1	2009 Audi A3 2.0 T AT	140	143	202	
2	2009 Audi A3 2.0 T	140	143	202	
3	2009 Audi A3 2.0 T Quattro	140	143	202	
4	2009 Audi A3 2.0 T Quattro	140	143	202	
..	…	…	…	…	
695	2012 Ford Mustang GT AT	137	169	85	
696	2012 Ford Mustang GT Premium	137	169	85	
697	2012 Ford Mustang GT Premium AT	137	169	85	
698	2012 Ford Mustang Boss 302	119	169	85	
699	2010 Mercedes-Benz C300 Luxury 4MATIC	165	22	234	

	Number of Forward Gears	City mpg	Highway mpg	Horsepower	Torque
0	6	18	25	250	236
1	6	22	28	200	207
2	6	21	30	200	207
3	6	21	28	200	207
4	6	21	28	200	207
..	…	…	…	…	…
695	6	18	25	412	390
696	6	17	26	412	390
697	6	18	25	412	390
698	6	17	26	444	380
699	7	18	25	228	221

[700 rows x 9 columns]

[33]:
```
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score

# training data
X = car[["Width"]].values
# training target
y = car["Horsepower"].values

r = 0
log = []
kf = KFold(n_splits=10)
for train, test in kf.split(X):
    x_train, x_test = X[train], X[test]
    y_train, y_test = y[train], y[test]

    theta = grad_descent_log(x_train, y_train, 10000, .0000001)


    if ((np.exp(x_test.dot(theta)) / (1 + np.exp(x_test.dot(theta))))).all() >=␣
 ↪0.5 ):
        r = 1
    else:
        r = 0
    log.append((r == y_test).mean())
log
```

```
<ipython-input-30-2104f33dfb80>:15: RuntimeWarning: overflow encountered in exp
  p = (np.exp(X.dot(theta)) / (1 + np.exp(X.dot(theta))))
<ipython-input-30-2104f33dfb80>:15: RuntimeWarning: invalid value encountered in
true_divide
  p = (np.exp(X.dot(theta)) / (1 + np.exp(X.dot(theta))))
```

[33]:
```
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
```

[34]:
```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn import svm

# cross-validation
# Linear Discriminant Analysis
lda = LinearDiscriminantAnalysis()
lda.fit(X, y)
lda_cvs = cross_val_score(lda, X, y, cv=10)
print(lda_cvs)

# Linear SVM
clf = svm.SVC()
clf.fit(X, y)
```

```
clf_cvs = cross_val_score(clf, X, y, cv=10)
print(clf_cvs)
```

```
/opt/conda/lib/python3.8/site-packages/sklearn/model_selection/_split.py:670:
UserWarning: The least populated class in y has only 1 members, which is less
than n_splits=10.
  warnings.warn(("The least populated class in y has only %d"
/opt/conda/lib/python3.8/site-packages/sklearn/model_selection/_split.py:670:
UserWarning: The least populated class in y has only 1 members, which is less
than n_splits=10.
  warnings.warn(("The least populated class in y has only %d"
```

```
[0.2        0.14285714 0.14285714 0.17142857 0.18571429 0.14285714
 0.18571429 0.2        0.18571429 0.17142857]
[0.24285714 0.2        0.21428571 0.17142857 0.18571429 0.17142857
 0.21428571 0.17142857 0.2        0.14285714]
```

[35]:
```
# t tests
import scipy.stats as stats

print(stats.ttest_rel(lda_cvs, log))
print(stats.ttest_rel(clf_cvs, log))
```

```
Ttest_relResult(statistic=23.98771519651975, pvalue=1.8171643936358359e-09)
Ttest_relResult(statistic=21.070525218110557, pvalue=5.729697852811392e-09)
```