

In [3]: `Image(filename =r'CoverSheet.png')`

Out[3]:

CCT College Dublin

Assessment Cover Page

Module Title:	Statistical Techniques for Data Analysis
Assessment Title:	CA2 – Inferential Statistics
Lecturer Name:	John O’Sullivan
Student Full Name:	Arthur Claudino Gomes de Assis
Student Number:	2023146
Assessment Due Date:	22/05/2023
Date of Submission:	21/05/2023

|

Declaration

By submitting this assessment, I confirm that I have read the CCT policy on Academic Misconduct and understand the implications of submitting work that is not my own or does not appropriately reference material taken from a third party or other source. I declare it to be my own work and that all material from third parties has been appropriately referenced. I further confirm that this work has not previously been submitted for assessment by myself or someone else in CCT College Dublin or any other higher education institution.

```
In [2]: #Importing basic lybraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import math

#Importing statistic libraries
from scipy import stats
import scipy.stats as ss
from scipy.stats import f_oneway
from statsmodels.stats.multicomp import pairwise_tukeyhsd
from statsmodels.formula.api import ols
import statsmodels.api as sm
from statsmodels.stats.anova import anova_lm

#Importing library to analyse correlation
from scipy.stats import pearsonr

#Importing library to deal with images
from skimage import io
from IPython.display import Image

#Importing library to perform regression
from sklearn.linear_model import LinearRegression
```

```
In [4]: def glimpse(df):
        print(f"The dataset has {df.shape[0]} observartions and {df.shape[1]} attributes")
        display(df.head())
        display(df.tail())
```

EXERCISE 1

```
In [5]: #Reading the dataset required by the exercise

df_Q1 = pd.read_csv("Q1.csv")
glimpse(df_Q1)
```

The dataset has 50 observartions and 1 attributes

	exam_score
0	74.54
1	66.20
2	86.75
3	70.87
4	80.43
	exam_score
45	61.74
46	58.77
47	49.62
48	71.77
49	83.08

```
In [6]: #Printing the descriptive statistics of the data

df_Q1.describe()
```

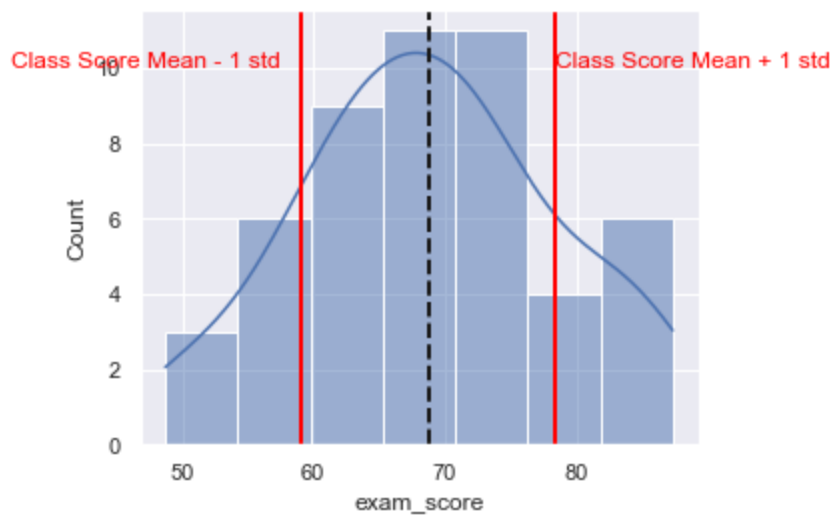
```
Out[6]:
```

	exam_score
count	50.000000
mean	68.735600
std	9.750143
min	48.730000
25%	62.040000
50%	68.390000
75%	74.630000
max	87.360000

```
In [7]: #Plotting an appropriate to summarise the dataset

fig = sns.set(rc = {'figure.figsize':(5,4)})
sns.histplot(data=df_Q1, x='exam_score', kde= True)
plt.axvline(df_Q1['exam_score'].mean(), color='k', linestyle='dashed', linewidth=2)
plt.axvline(df_Q1['exam_score'].mean()+np.std(df_Q1['exam_score']), color='red', linestyle='solid', linewidth=2)
plt.axvline(df_Q1['exam_score'].mean()-np.std(df_Q1['exam_score']), color='red', linestyle='solid', linewidth=2)
plt.text(df_Q1['exam_score'].mean()+np.std(df_Q1['exam_score']), 10, "Class Score Mean + 1 std", color='red')
plt.text(37, 10, 'Class Score Mean - 1 std', color = 'red')
```

```
Out[7]: Text(37, 10, 'Class Score Mean - 1 std')
```



It is possible to verify that the sample of 50 students has an exam scores mean of 68.73, which is below the national average. It is also possible to verify that the sample standard deviation is 9.75. Although the mean of the sample is the best point estimation, and it is below the national mean, we cannot conclude without further statistic analysis that the average of the population is below the national average. To verify it, it will be performed an hypothesis test. Once we are interested in discovering whether the mean of the population is below 70, which is the national average, it will be performed a one sample left-tail test. The level of significance in this case is $\alpha = 0.05$.

Stating our hypothesis

Null hypothesis: The performance of the students of the school is equal to the national average.

Alternative hypothesis: The performance of the students of the school is below the national average.

H0: $\mu = 70$

HA: $\mu < 70$

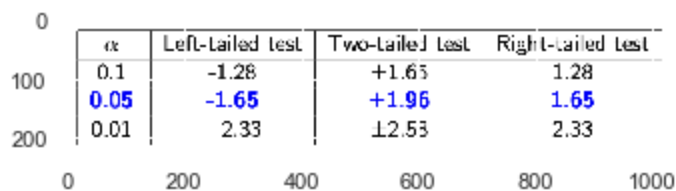
Finding a critical value to define rejection region

If $p\text{-value} < 0.05$ we reject H0 in favour of HA and if $p\text{-value} \geq 0.05$ we fail to reject H0, therefore we have no evidence enough in favour of HA.

The values for choosing the rejection region is presented in the table below.

```
In [8]: img = io.imread('Figure 1.png')
        io.imshow(img)
```

```
Out[8]: <matplotlib.image.AxesImage at 0x7f7b6a05d2e0>
```



α	Left-tailed test	Two-tailed test	Right-tailed test
0.1	-1.28	+1.65	1.28
0.05	-1.65	+1.96	1.65
0.01	2.33	2.53	2.33

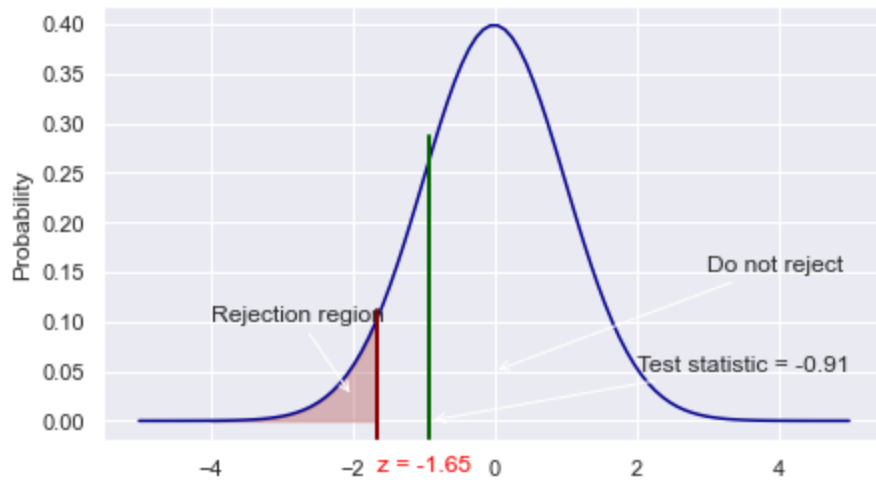
```
In [9]: stats.ttest_1samp(df_Q1, popmean=70, alternative = "less")
```

```
Out[9]: Ttest_1sampResult(statistic=array([-0.91697714]), pvalue=array([0.18182299]))
```

```
In [10]: x = np.linspace(-5, 5, 100)
         y = ss.norm.pdf(x)
```

```
In [11]: fig, ax = plt.subplots(figsize=(7,4))
         ax.plot(x,y, color='darkblue')
         plt.axvline(-1.65, 0, 0.30, color='darkred', linestyle="solid", linewidth=2)
         plt.text(-1.65, -0.05, "z = -1.65", color = 'red')
         plt.axvline(-0.91, 0, 0.70, color='darkgreen', linestyle="solid", linewidth=2)
         plt.annotate(text='Test statistic = -0.91', xy=(-0.91,0), xytext=(2,0.05), arrowprops=dict(
         plt.annotate(text='Rejection region', xy=(-2,0.025), xytext=(-4,0.1), arrowprops=dict(arrow
         plt.annotate(text='Do not reject', xy=(0,0.05), xytext=(3,0.15), arrowprops=dict(arrowsty
         x_fill = np.linspace(-4, -1.65, 100)
         y_fill= ss.norm.pdf(x_fill, 0, 1)
         ax.fill_between(x_fill, y_fill, alpha=0.3, color='brown')
         ax.set_ylabel('Probability')
```

```
Out[11]: Text(0, 0.5, 'Probability')
```



As it is possible to verify, the one-sample left-tailed test returned a test statistic of -0.91, while the rejection region for a level of significance of 0.05 is below -1.65, therefore we can conclude that the value is in the region in which we fail to reject our null hypothesis. This result is also confirmed through the p-value. p-value > 0.05 , so we do not have enough evidence to reject the null hypothesis, in other words, we fail to reject it. To conclude, there is no evidence enough to support the teacher's concerns.

EXERCISE 2

In [12]:

```
#Import package and reading the dataset diamonds

from pydataset import data
df_diamonds = data('diamonds')

#Printing the first 5 rows of the dataset

df_diamonds.head(5)
```

Out[12]:

	carat	cut	color	clarity	depth	table	price	x	y	z
1	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
2	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
3	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
4	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
5	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75

Color from D to F are considered colourless, therefore they are considered to have better quality. The column colourless will be considered 1 for these values.

Color from G to J are not considered colourless. The columns colourless will be considered 0 for these values.

In [13]:

```
#Creating a new binary variable in the dataframe named colourless to record colourless and non colourless diamonds

df_diamonds['colourless'] = df_diamonds['color'].apply(lambda x: 1 if (x == 'D' or x == 'E'
```

In [14]:

```
#Printing the dataframe with the new variable
```

```
df_diamonds.head()
```

```
Out[14]:
```

	carat	cut	color	clarity	depth	table	price	x	y	z	colourless
1	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43	1
2	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31	1
3	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31	1
4	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63	0
5	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75	0

```
In [15]: #Verifying the values on the column clarity

df_diamonds.clarity.unique()
```

```
Out[15]: array(['SI2', 'SI1', 'VS1', 'VS2', 'VVS2', 'VVS1', 'I1', 'IF'],
              dtype=object)
```

```
In [16]: # Making a two-way contingency table:

clarity_colourless = pd.pivot_table(df_diamonds, index=['clarity'], columns=['colourless'],
                                     clarity_colourless
```

```
Out[16]:
```

	colourless	0	1
clarity			
I1	454	287	
IF	1174	616	
SI1	6425	6640	
SI2	4502	4692	
VS1	4821	3350	
VS2	5890	6368	
VVS1	2013	1642	
VVS2	2547	2519	

The pivot table was created so that every classification of clarity of the diamonds are counted for being colourless and non-colourless. For instance, the clarity SI2 has 4502 diamonds classified as non colourless and 4692 diamonds classified as colourless. We are interest in discover whether there is any correlation between the variables Colourless and Clarity, with a level of significance $\alpha = 0.01$. Therefore, our hypothesis test will be resumed as:

Stating our hypothesis

H0: Colourless and Clarity are independent

HA: Colourless and Clarity are not independent

If p-value < 0.01 we reject H0 in favour of Ha.

If p-values ≤ 0.01 we do not reject H_0 .

The test able to perform an analysis between two categorical variables, clarity and colourless, which is a boolean variable, is a two sample χ^2 test.

In [17]:

```
#Performing a chi2 test to verify whether the variables are independent or not

chi2_stat, p_val, dof, ex = stats.chi2_contingency(clarity_colourless)

print("Chi square value is ",chi2_stat)
print("P value is",p_val)
print("Degrees of Freedom:",dof)
```

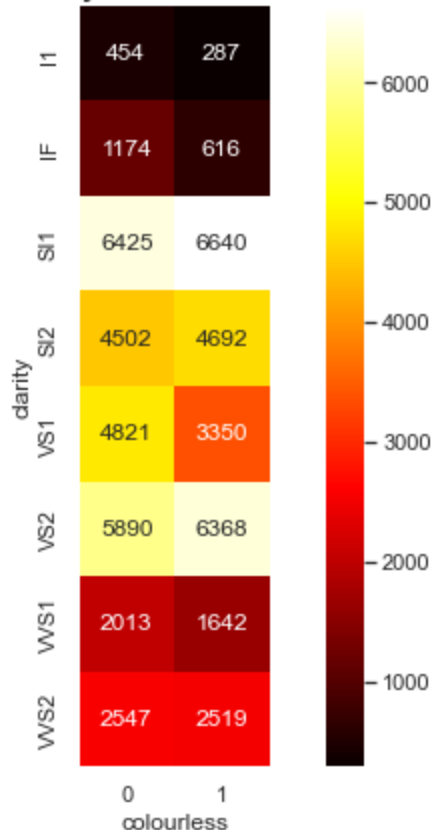
```
Chi square value is  486.47305941485223
P value is 6.481587124454715e-101
Degrees of Freedom:  7
```

It is possible to verify that p-value is virtually 0, therefore it is lesser than 0.01. Thus we can reject our null hypothesis in favour of an alternative hypothesis with confidence, and we can conclude we have enough evidence to consider that Colourless and Clarity are not independent. In other words, whether or not a diamond is colourless depends on its classification of Clarity.

In [18]:

```
# Plotting a heatmap
fig1= sns.set(rc = {'figure.figsize':(10,7)})
sns.heatmap(clarity_colourless, annot=True, fmt='g',square=True, cmap='hot')
plt.title('Clarity Vs Colourless',fontsize=20)
plt.show()
```

Clarity Vs Colourless



In [19]:

```
#Plotting a graph to visualize proportion of colourless and non colourless diamonds for each clarity category

clarity_colourlessmelt = pd.melt(clarity_colourless.reset_index(), id_vars='clarity')
```



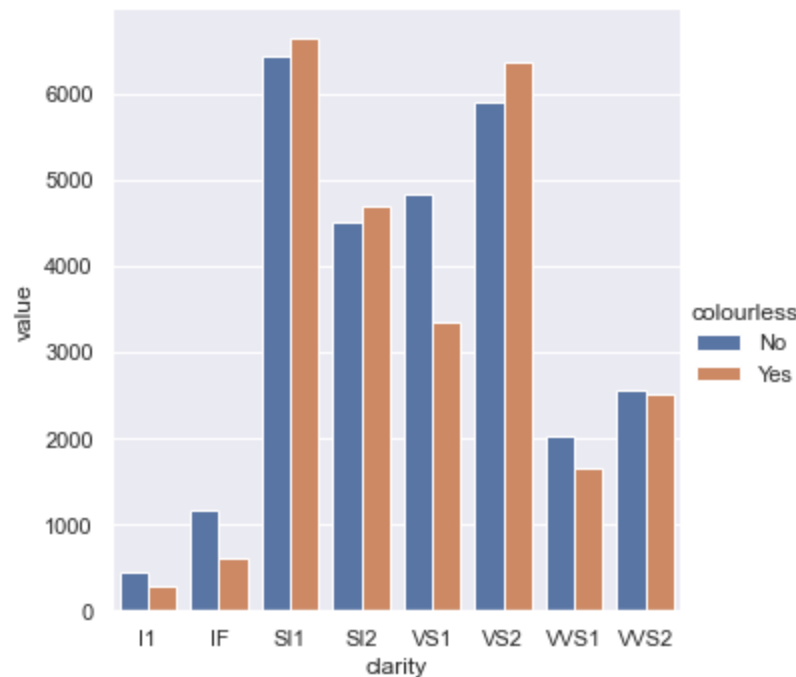
```

clarity_colourlessmelt['colourless'].replace({1: 'Yes', 0: 'No'}, inplace = True)
print(clarity_colourlessmelt)

sns.set(rc = {'figure.figsize':(10,4)})
sns.catplot(x = 'clarity', y='value',
            hue = 'colourless',data=clarity_colourlessmelt, kind='bar')
plt.show()

```

	clarity	colourless	value
0	I1	No	454
1	IF	No	1174
2	SI1	No	6425
3	SI2	No	4502
4	VS1	No	4821
5	VS2	No	5890
6	VVS1	No	2013
7	VVS2	No	2547
8	I1	Yes	287
9	IF	Yes	616
10	SI1	Yes	6640
11	SI2	Yes	4692
12	VS1	Yes	3350
13	VS2	Yes	6368
14	VVS1	Yes	1642
15	VVS2	Yes	2519



```
In [20]: df_diamonds.shape
```

```
Out[20]: (53940, 11)
```

```
In [21]: print('Proportion of diamonds non colourless: ', df_diamonds[df_diamonds['colourless']==0].shape[0]/df_diamonds.shape[0])
print('Proportion of diamonds colourless: ', df_diamonds[df_diamonds['colourless']==1].shape[0]/df_diamonds.shape[0])
```

```

Proportion of diamonds non colourless:  [0.51586948]
Proportion of diamonds colourless:  [0.48413052]

```

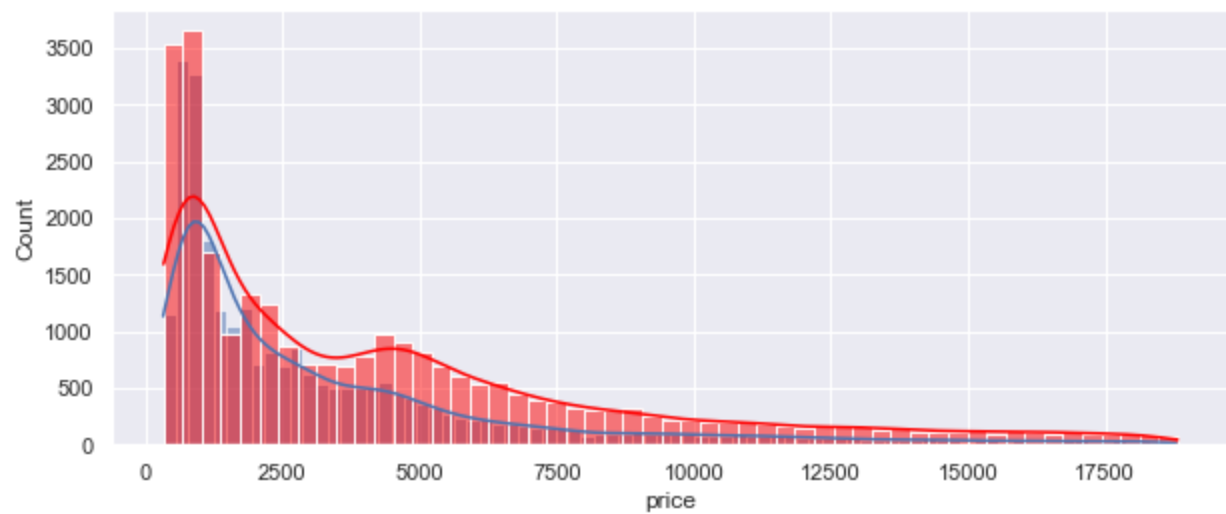
It is possible to verify that our dataset contains information of 53940 diamonds, in a proportion of 51.58% non colourless and 48.41% colourless. This means that if we had failed to reject our null hypothesis, we should have proportions near to that in every classification of clarity. The barplot shows that the proportion of colourless and non colourless diamonds varies significantly depending on the clarity. For instance, when

comparing VS1, SI1 and VS2, we verify that the proportion (non colourless/colourless) is greater than 1 in the former, whereas it is lesser than 1 in the other two.

Although the p-value of the χ^2 test is the most reliable evidence of our failure in rejecting our null hypothesis, observing the graph also provide strong evidence that the variables clarity and colourless cannot be considered independet.

Analysing the price of the diamonds

```
In [22]: #Plotting overlapping histplots for prices of colourless and non colourless diamonds
sns.set(rc = {'figure.figsize':(10,4)})
fig1 = sns.histplot(data=df_diamonds[df_diamonds['colourless']==1], x='price', kde= True)
fig1 = sns.histplot(data=df_diamonds[df_diamonds['colourless']==0], x='price', kde= True,
#fig1.set(xlabel='price')
```



```
In [23]: #Verifying descriptive analysis of colourless diamonds

df_diamonds[df_diamonds['colourless']==1].describe()
```

	carat	depth	table	price	x	y	z	c
count	26114.000000	26114.000000	26114.000000	26114.000000	26114.000000	26114.000000	26114.000000	
mean	0.686595	61.683312	57.44766	3337.759401	5.487314	5.492809	3.386464	
std	0.379068	1.433278	2.24200	3527.047348	0.978025	0.985360	0.629647	
min	0.200000	51.000000	44.00000	326.000000	0.000000	0.000000	0.000000	
25%	0.380000	61.000000	56.00000	919.000000	4.640000	4.650000	2.850000	
50%	0.560000	61.800000	57.00000	1938.500000	5.310000	5.320000	3.270000	
75%	1.000000	62.500000	59.00000	4372.000000	6.300000	6.300000	3.890000	
max	3.400000	79.000000	95.00000	18791.000000	9.420000	31.800000	31.800000	

```
In [24]: #Verifying descriptive analysis of non colourless diamonds

df_diamonds[df_diamonds['colourless']==0].describe()
```

	carat	depth	table	price	x	y
count	27826.000000	27826.000000	27826.000000	27826.000000	27826.000000	27826.000000

	carat	depth	table	price	x	y	
mean	0.902434	61.811432	57.466122	4491.230073	5.959997	5.961371	3.68163
std	0.527400	1.429253	2.227423	4305.086420	1.197273	1.229325	0.74239
min	0.230000	43.000000	43.000000	334.000000	0.000000	0.000000	0.00000
25%	0.410000	61.100000	56.000000	998.000000	4.790000	4.800000	2.96000
50%	0.840000	61.900000	57.000000	3070.000000	6.020000	6.010000	3.73000
75%	1.200000	62.600000	59.000000	6334.000000	6.810000	6.810000	4.21000
max	5.010000	73.600000	76.000000	18823.000000	10.740000	58.900000	8.06000

In [25]:

```
#Printing the mean, standard deviation and lenght of colourless diamonds subset

print('The mean price of colourless diamonds is: ', df_diamonds['price'][df_diamonds['color']=='colorless'].mean())
print('The standard deviation of colourless diamonds is: ', np.std(df_diamonds['price'][df_diamonds['color']=='colorless']))
print('There are {} samples of colourless diamonds.'.format(df_diamonds['price'][df_diamonds['color']=='colorless'].count()))
```

The mean price of colourless diamonds is: 3337.759401087539
The standard deviation of colourless diamonds is: 3526.979815359925
There are 26114 samples of colourless diamonds.

In [26]:

```
#Printing the mean, standard deviation and lenght of non colourless diamonds subset

print('The mean price of not colourless diamonds is: ', df_diamonds['price'][df_diamonds['color']!='colorless'].mean())
print('The standard deviation of not colourless diamonds is: ', np.std(df_diamonds['price'][df_diamonds['color']!='colorless']))
print('There are {} samples of not colourless diamonds.'.format(df_diamonds['price'][df_diamonds['color']!='colorless'].count()))
```

The mean price of not colourless diamonds is: 4491.230072593977
The standard deviation of not colourless diamonds is: 4305.009062400536
There are 27826 samples of not colourless diamonds.

In [27]:

```
img = io.imread('Figure 2.png')
io.imshow(img)
print('Values to be applied when calculating intervals of confidence using z-distribution')
```

Values to be applied when calculating intervals of confidence using z-distribution

	Confidence Interval	$\frac{\alpha}{2}$	$z_{1-\frac{\alpha}{2}}$
0			
50			
100	90%	0.05	1.645
150	95%	0.025	1.96
200	99%	0.005	2.58
250			
0			
100			
200			
300			
400			
500			
600			

Interval estimate:

We have data of 53940 diamonds, however, we do not have information whether this is a sample or the whole population that should be analysed for diamonds. Once we do not have the information of the the

standard deviation of the whole population, the confidence interval will be calculating using a t-distribution.

```
In [28]: # Calculating a 90% confidence interval to colourless diamonds

interval_colourless = stats.t.interval(0.90,
                                       len(df_diamonds['price'][df_diamonds['colourless']==1]),
                                       loc = (df_diamonds['price'][df_diamonds['colourless']==1]).mean(),
                                       scale = np.std(df_diamonds['price'][df_diamonds['colourless']==1])/np.sqrt(len(df_diamonds['price'][df_diamonds['colourless']==1]))
```

```
In [29]: # Calculating a 90% confidence interval to non colourless diamonds

interval_noncolourless = stats.t.interval(0.90,
                                       len(df_diamonds['price'][df_diamonds['colourless']==0]),
                                       loc = (df_diamonds['price'][df_diamonds['colourless']==0]).mean(),
                                       scale = np.std(df_diamonds['price'][df_diamonds['colourless']==0])/np.sqrt(len(df_diamonds['price'][df_diamonds['colourless']==0]))
```

```
In [30]: print('The 90% confidence interval for colourless diamonds is: ', interval_colourless)
```

The 90% confidence interval for colourless diamonds is: (3301.8582032924432, 3373.660598882635)

```
In [31]: print('The 90% confidence interval for non-colourless diamonds is: ', interval_noncolourless)
```

The 90% confidence interval for non-colourless diamonds is: (4448.778831103339, 4533.681314084614)

As it is observed, the price of the colourless diamonds are considerably smaller than the price of non colourless diamonds, in despite of being considered of higher quality. However, when we compare the descriptive analysis of both kind of diamonds, we verify that the feature 'carat' of non colourless diamonds is much higher in average than the same feature of colourless diamonds, this may account for the highest difference in price.

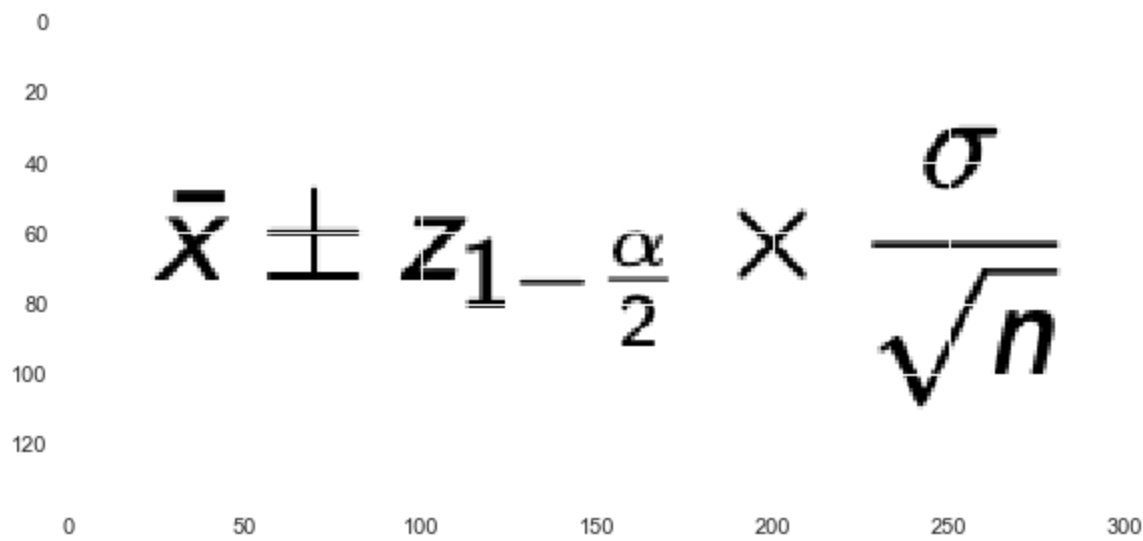
We can conclude that we have 90% of confidence that the mean of price of colourless diamonds are between 3301.85 and 3373.66, and that the mean of price of non-colourless diamonds are between 4448.79 and 4533.68.

In case we wanted to calculate que confidence intervals considering the dataset is the whole population, we would use the formula below to do it, in which $(z-\alpha/2)$ is given on the table above, and it is equal to 1.645 for 90% of confidence. The calculus are shown as is follows.

It is possible to verify that the difference using both the normal-distribution and the t-distribution is very small. This happens for the amount of observations analysed, for a high amount of observations the t-distribution behaves similarly to a normal distribution, and therefore, similar results are expected.

```
In [32]: img = io.imread('Figure 3.png')
io.imshow(img)
print('Formula used to calculate confidence intervals using a normal distribution')
```

Formula used to calculate confidence intervals using a normal distribution



In [33]:

```
#Calculating the confidence interval according to the formula above

interval_colourless2 = 1.645*np.std(df_diamonds['price'][df_diamonds['colourless']==1])/ma
interval_noncolourless2 = 1.645*np.std(df_diamonds['price'][df_diamonds['colourless']==0])
print('The confidence interval of colourless diamonds is: ({} , {})' .format(df_diamonds['pr
df_diamonds['pr

print('The confidence interval of colourless diamonds is: ({} , {})' .format(df_diamonds['pr
df_diamonds['pr

The confidence interval of colourless diamonds is: (3301.856282244873,3373.6625199302052)
The confidence interval of colourless diamonds is: (4448.776466898911,4533.683678289042)
```

EXERCISE 3

In [34]:

```
#Reading the dataset PlantGrowth

df_plant = data('PlantGrowth')
glimpse(df_plant)
```

The dataset has 30 observartions and 2 attributes

	weight	group
1	4.17	ctrl
2	5.58	ctrl
3	5.18	ctrl
4	6.11	ctrl
5	4.50	ctrl
	weight	group
26	5.29	trt2
27	4.92	trt2
28	6.15	trt2
29	5.80	trt2
30	5.26	trt2

In [35]:

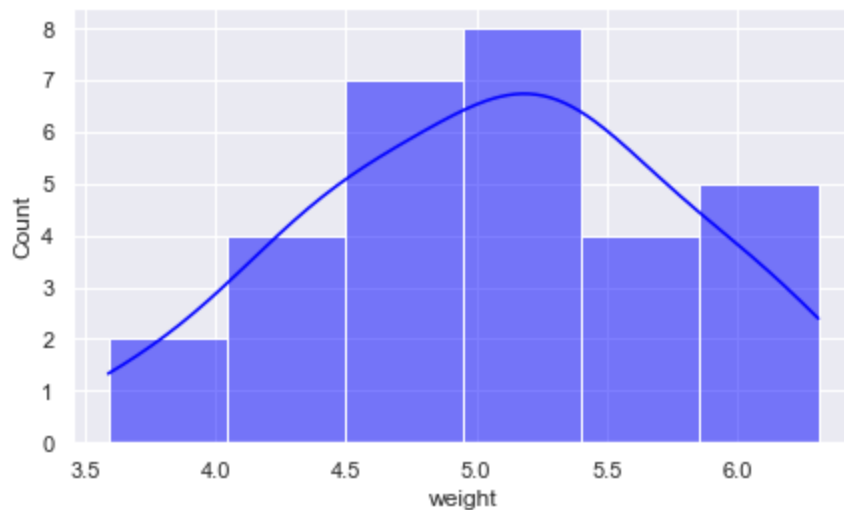
```
#Presenting the descriptive analysis of the whole dataset
```

```
df_plant.describe()
```

```
Out[35]:
```

	weight
count	30.000000
mean	5.073000
std	0.701192
min	3.590000
25%	4.550000
50%	5.155000
75%	5.530000
max	6.310000

```
In [36]: #Plotting a histogram of the whole dataset, with all its categories.  
sns.set(rc = {'figure.figsize':(7,4)})  
fig1 = sns.histplot(data=df_plant, x='weight', kde= True, color = 'blue')
```

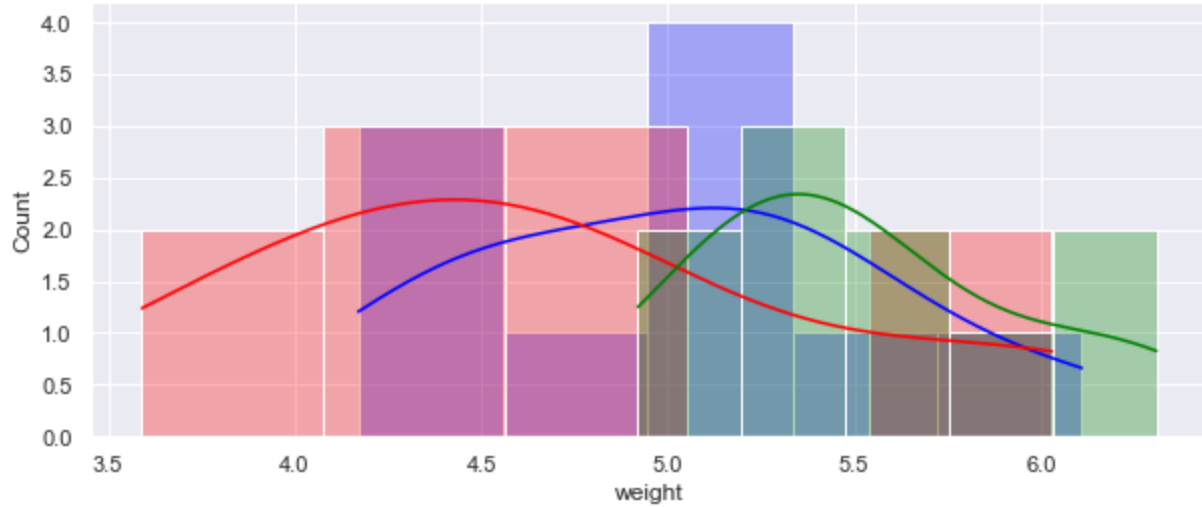


```
In [37]: #Verifying the categories on the feature group to plot them separately  
df_plant['group'].unique()
```

```
Out[37]: array(['ctrl', 'trt1', 'trt2'], dtype=object)
```

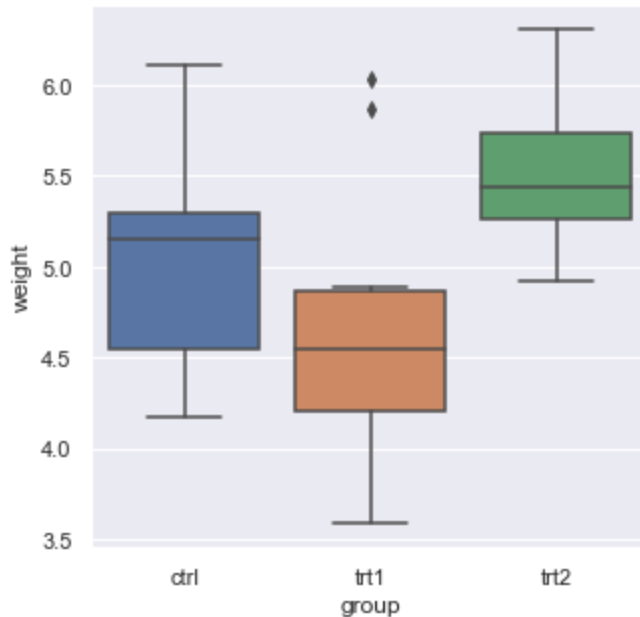
```
In [38]: #Plotting a histogram with separate categories of the fature group  
  
fig1= sns.set(rc = {'figure.figsize':(10,4)})  
fig1 = sns.histplot(data=df_plant[df_plant['group']=='ctrl'], x='weight', kde= True, color= 'red')  
fig1 = sns.histplot(data=df_plant[df_plant['group']=='trt1'], x='weight', kde= True, color= 'green')  
fig1 = sns.histplot(data=df_plant[df_plant['group']=='trt2'], x='weight', kde= True, color= 'blue')  
fig1.set(xlabel='weight')
```

```
Out[38]: [Text(0.5, 0, 'weight')]
```



```
In [39]: #Plotting a Boxplot split by categories on the feature group
fig = sns.set(rc = {'figure.figsize':(5,5)})
sns.boxplot(data = df_plant, x='group', y='weight')
```

```
Out[39]: <AxesSubplot:xlabel='group', ylabel='weight'>
```



```
In [40]: #Creating 3 subset from the original dataset to develop statistical tests on them
```

```
df_plant_ctrl = df_plant[df_plant['group']=='ctrl']
df_plant_trt1 = df_plant[df_plant['group']=='trt1']
df_plant_trt2 = df_plant[df_plant['group']=='trt2']
```

```
In [41]: #Source: https://stackoverflow.com/questions/38783027/jupyter-notebook-display-two-pandas-
#Displaying multiple table side by side
```

```
from IPython.display import display_html
from itertools import chain,cycle
def display_side_by_side(*args,titles=cycle([''])):
    html_str=''
    for df,title in zip(args, chain(titles,cycle(['<br>'])) ):
        html_str+<th style="text-align:center"><td style="vertical-align:top">'
        html_str+f<h2 style="text-align: center;">{title}</h2>'
        html_str+=df.to_html().replace('table','table style="display:inline"')
```

```
html_str+='\n</td></th>'\n\ndisplay_html(html_str,row=True)
```

```
In [42]: display_side_by_side(df_plant_ctrl.describe(), df_plant_trt1.describe(), df_plant_trt2.des\n                                titles = ['ctrl Description', 'trt1 Description', 'trt2 Description'])\nprint('Figure generated using method from Stack Overflow, n.d.')
```

ctrl Description

	weight
count	10.000000
mean	5.032000
std	0.583091
min	4.170000
25%	4.550000
50%	5.155000
75%	5.292500
max	6.110000

trt1 Description

	weight
count	10.000000
mean	4.661000
std	0.793676
min	3.590000
25%	4.207500
50%	4.550000
75%	4.870000
max	6.030000

trt2 Description

	weight
count	10.000000
mean	5.526000
std	0.442573
min	4.920000
25%	5.267500
50%	5.435000
75%	5.735000
max	6.310000

Figure generated using method from Stack Overflow, n.d.

In [43]:

```
interval_weight = stats.t.interval(0.99, len(df_plant['weight'])-1,
                                   loc = df_plant['weight'].mean(),
                                   scale=np.std(df_plant['weight'])/np.sqrt(len
print(interval_weight)

(4.726059804043951, 5.419940195956052)
```

It is possible to verify that the dataframe contains balanced observations in regards to plants under control and under both of treatment conditions, being 10 observations for each. The mean of the weight in the whole dataset is 5.07 and its standard deviation is 0.70. Values of weight are within an interval (4.7260, 5.4199) with a level of significance of 99%. The weight seems to follow a distribution nearly normal, slightly negatively skewed. However, this observation is merely visual and it would require a statistical test to be confirmed.

When verifying the different variables influencing the plants, it is possible to verify the control test in an intermediate position of both mean and standard deviation. Treatment 1 seems to reduce the mean of weight and increase its standard deviation, which in case one aimed to increase weight through treatment, could lead to a conclusion at first sight that this treatment had negative results. On the other hand, treatment 2 has an increased weight average and reduced standard deviation. This information can be verified through the histogram and boxplot.

Although these are visual conclusions drawn from point estimation, to reach a reliable conclusion in regards to the efficiency of the treatments according to the data analysed, it is necessary to perform a hypothesis test, to analyse if in fact, the differences observed through this 30 observations sample, could be extrapolated to the population.

The test performed to verify difference of means in samples is the ANOVA. In this case we are looking for a significance level $\alpha=0.05$. Our test of hypothesis will be as it follows:

Stating our hypothesis:

H0 : Mean_ctrl = Mean_trt1 = Mean_trt2 - There is no evidence enough that the means differ.

HA : There is evidence enough that at least one mean differs from the rest.

If p-value < 0.05 we reject H0 in favour of HA.

If p-values ≥ 0.05 we do not reject H0.

In [44]:

```
# Performing the one-way ANOVA test:
```

```
stats.f_oneway(df_plant_ctrl['weight'], df_plant_trt1['weight'], df_plant_trt2['weight'])
```

```
Out[44]: F_onewayResult(statistic=4.846087862380136, pvalue=0.0159099583256229)
```

It is possible to verify that $p\text{-value} < 0.05$, therefore we reject H_0 in favour of H_A . In other words, there is evidence enough of difference on the means. This difference can be again observed through the histogram and boxplot, moreover at this time it is confirmed by the ANOVA test.

To verify where this difference comes from, it is necessary to perform a Tukey-Kramer test.

```
In [45]: significance_level = 0.05

# Perform Tukey-Kramer Analysis:

tukey = pairwise_tukeyhsd(endog=df_plant['weight'],
                           groups=df_plant['group'],
                           alpha=significance_level)

#display results
print(tukey)
```

```
Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====
group1 group2 meandiff p-adj    lower    upper  reject
-----
ctrl    trt1    -0.371 0.3909 -1.0622  0.3202   False
ctrl    trt2     0.494 0.198  -0.1972  1.1852   False
trt1    trt2     0.865 0.012   0.1738  1.5562    True
-----
```

As it is possible to verify with the Tukey-Kramer test, we fail to reject H_0 when comparing the control group with both treatment 1 and 2, however, when comparing treatment 1 and 2, we have enough evidence to consider they have different means. This result can be observed in the boxplot, where we can verify overlapping regions between control and both treatments, but there are no overlapping regions between both treatments.

We can conclude that in despite of not having evidence of the effectiveness of treatment 1 and 2 in relation to the control group, we have evidence that treatment 1 and 2 produce different effects in the weight of the plants.

EXERCISE 4

```
In [46]: #Reading the dataset trees

df_trees = data('trees')
glimpse(df_trees)
```

The dataset has 31 observations and 3 attributes

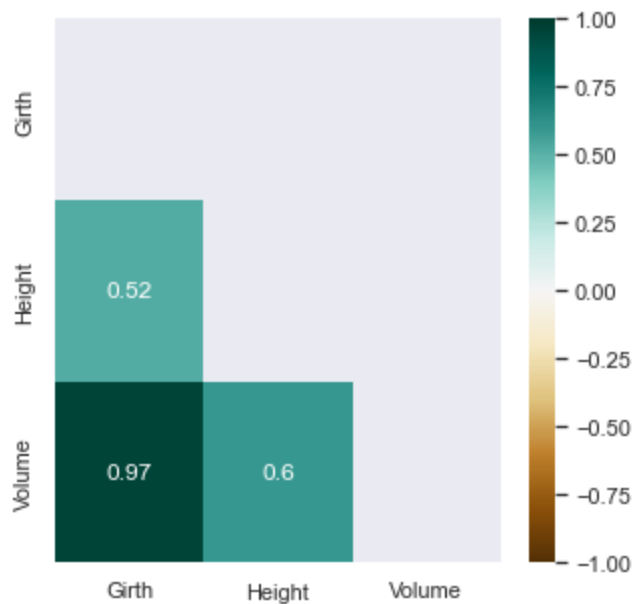
	Girth	Height	Volume
1	8.3	70	10.3
2	8.6	65	10.3
3	8.8	63	10.2
4	10.5	72	16.4
5	10.7	81	18.8

	Girth	Height	Volume
27	17.5	82	55.7
28	17.9	80	58.3
29	18.0	80	51.5
30	18.0	80	51.0
31	20.6	87	77.0

In [47]:

```
#Plotting a correlation matrix to verify summary analysis
```

```
corr_matrix = df_trees.corr()
mask = np.triu(np.ones_like(corr_matrix))
sns.heatmap(corr_matrix, annot=True, vmax=1, vmin=-1, center=0, cmap='BrBG', mask=mask)
plt.show()
```



It is possible to verify the correlation coefficients among the three different variables on the dataset. What we aim now is to identify if this correlations indicates indeed relationship between variables, what considers, for instance, the size of the dataset. We want to verify this information with a level of confidence $\alpha = 0.01$.

Stating our hypothesis

H0: Correlation equals to 0.

HA: Correlation does not equal to 0.

If p-value < 0.01 we reject H0 in favour of HA.

If p-values ≥ 0.01 we fail to reject H0.

In [48]:

```
stats.pearsonr(df_trees['Girth'], df_trees['Height'])
```

Out[48]:

```
(0.5192800719499373, 0.002757814793057672)
```

In [49]:

```
stats.pearsonr(df_trees['Girth'], df_trees['Volume'])
```

Out[49]: (0.9671193682556306, 8.644334211770157e-19)

In [50]: `stats.pearsonr(df_trees['Height'], df_trees['Volume'])`

Out[50]: (0.5982496519917821, 0.0003783823479184879)

We can verify through the personr tests that, even with a level of confidence of 0.01, the p-value for all the pairs of variables on the dataset are lesser than α . It means that in every case we reject the null hypothesis, and we have enough evidence to consider that there is a correlation between all variables in the dataset. Moreover, we can verify that Girth and Volume have a strong correlation, with a statistic coefficient of 0.9671, whereas the other two have a weaker correlation.

In [51]: `#Creating a dataframe to store results of tests and a model to perform linear regressions`
`results= pd.DataFrame(columns = ['test', 'R2'])`
`model = LinearRegression()`

In [52]: `#Performing a LinearRegression using both Girth and Height`

`#The dependent variable X will drop only the dependent variable 'Volume'`
`X = df_trees.drop(columns = ['Volume'])`
`y = df_trees['Volume']`

`model1 = model.fit(X, y)`
`cv_results = model1.score(X,y)`
`print(cv_results)`

0.9479500377816745

In [53]: `#Saving the result in the dataframe 'results'`
`results = results.append({'test': 'Girth&Height', 'R2': cv_results}, ignore_index = True)`

In [54]: `#Performing a LinearRegression using only Girth`

`#The dependent variable X will drop the dependent variable 'Volume' and the variable 'Height'`
`X = df_trees.drop(columns = ['Volume', 'Height'])`
`y = df_trees['Volume']`

`model2 = model.fit(X, y)`
`cv_results = model2.score(X,y)`
`print(cv_results)`

0.9353198724551699

In [55]: `#Saving the result in the dataframe 'results'`
`results = results.append({'test': 'Girth', 'R2': cv_results}, ignore_index = True)`

In [56]: `#Performing a LinearRegression using only Height`

`#The dependent variable X will drop the dependent variable 'Volume' and the variable 'Girth'`
`X = df_trees.drop(columns = ['Volume', 'Girth'])`
`y = df_trees['Volume']`

`model3 = model.fit(X, y)`

```
cv_results = model3.score(X,y)
print(cv_results)
```

0.3579026461082888

```
In [57]: results = results.append({'test': 'Height', 'R2': cv_results}, ignore_index = True)
```

```
In [58]: results.head()
```

```
Out[58]:
```

	test	R2
0	Girth&Height	0.947950
1	Girth	0.935320
2	Height	0.357903

It is possible to verify that the value of R2 improves significantly from using only 'Height' as independent variable to both using only 'Girth' and 'Girth and Height'. However, the difference between the models using only 'Girth' and using 'Girth and Height' is subtle.(Codecademy, n.d.) explains that comparing nested models using strictly R2 might be problematic, because it will always favor the larger and more complex model. Therefore it suggests using an adjusted R2, which penalises R2 to each additional predictor. This raises a hypothesis that adding a new variable to this model may, in fact, not improve the predictive power of the model. To verify this hypothesis it will be performed a F-test to evaluate whether using two variables to predict volume is better than using only one. We are going to use a confidence level $\alpha = 0.05$. Using this information we can state our hypothesis.

Stating our hypothesis

H0 : The new variable added sums up no more predictive power. All the coefficients of added features are equal to 0.

HA : The variables added sums up considerable predictive power.

If p-value < 0.05 we reject H0 in favour of HA.

If p-value \geq 0.05 we fail to reject H0.

```
In [59]: #Fitting the model considering only Girth as predictor

model1 = ols('Volume ~ Girth', data = df_trees).fit()
```

```
In [60]: #Fitting the model considering Girth and Height as predictor

model2 = ols('Volume ~ Girth + Height', data = df_trees).fit()
```

```
In [61]: #Performing an F-test using ANOVA to verify the improvement on the unrestricted model in c
#to the restricted

anova_results = anova_lm(model1, model2)
print(anova_results)
```

	df_resid	ssr	df_diff	ss_diff	F	Pr(>F)
--	----------	-----	---------	---------	---	--------

0	29.0	524.302539	0.0	NaN	NaN	NaN
1	28.0	421.921359	1.0	102.381179	6.79433	0.014491

```
In [62]: #Fitting the model considering only Height as predictor
```

```
model3 = ols('Volume ~ Height', data = df_trees).fit()
```

```
In [63]: #Fitting the model considering Height and Girth as predictor
```

```
model4 = ols('Volume ~ Height + Girth', data = df_trees).fit()
```

```
In [64]: anova_results = anova_lm(model3, model4)
print(anova_results)
```

	df_resid	ssr	df_diff	ss_diff	F	Pr(>F)
0	29.0	5204.895004	0.0	NaN	NaN	NaN
1	28.0	421.921359	1.0	4782.973645	317.412852	8.223304e-17

It is possible to verify from the table printed above comparing model 1 and 2 a p-value = 0.0145. Once p-value is smaller than $\alpha = 0.05$, we reject our null hypothesis in favour of an alternative hypothesis. Therefore we can consider that adding Height to our regression adds considerable predictive power. It is important to notice that p-value is somewhat high, and changing our confidence level to $\alpha = 0.01$ would lead to a different conclusion. This indicates that the predictive power of Height is not very strong. Tests were performed following methodology explored by ProjectPro, n.d..

On the other hand, when we make the same test analysing the improvement of a model containing only 'Height' and adding on 'Girth' (models 3 and 4, respectively), we notice that the p-value drops down to 8.223e-17, what is very near to 0. In this case we could reject our null hypothesis with much more confidence and argument that clearly adding 'Girth' to the model improves it significantly, as it is expected according to the R2 calculated before.

REFERENCES

1. Codecademy. (n.d.). Linear Regression in Python: Choosing a Linear Regression Model Cheatsheet. [online] Available at: <https://www.codecademy.com/learn/linear-regression-mssp/modules/choosing-a-linear-regression-model-mssp/cheatsheet>.
2. Stack Overflow. (n.d.). Jupyter notebook display two pandas tables side by side. [online] Available at: <https://stackoverflow.com/questions/38783027/jupyter-notebook-display-two-pandas-tables-side-by-side>.
3. ProjectPro. (n.d.). How to perform ANOVA using the StatsModels library in python? -. [online] Available at: <https://www.projectpro.io/recipes/perform-anova-statsmodels-library-python> [Accessed 21 May 2023].
4. Time Series Analysis, Regression, and Forecasting. (2022). Model Selection Tests For Nested And Non-nested Regression Models. [online] Available at: <https://timeseriesreasoning.com/contents/model-selection-tests-for-regression-models/> [Accessed 21 May 2023].