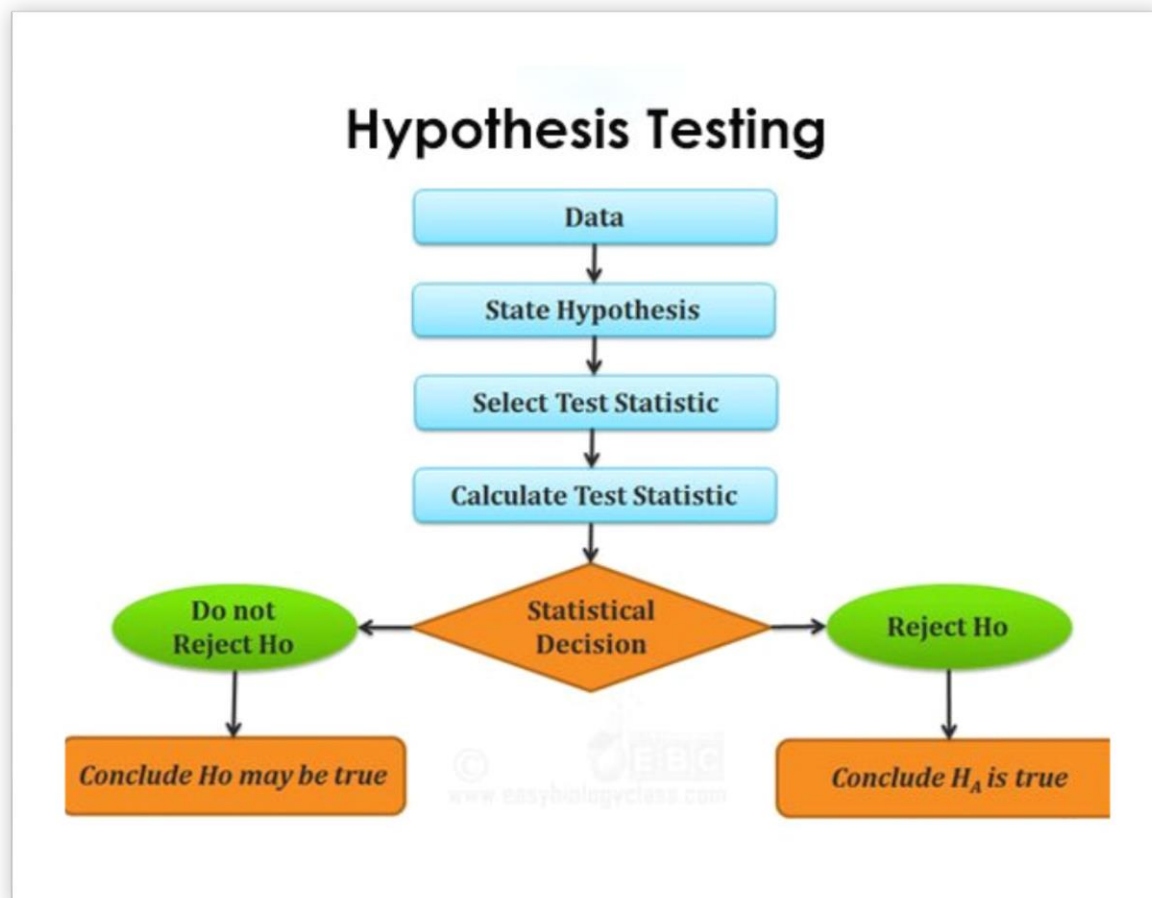


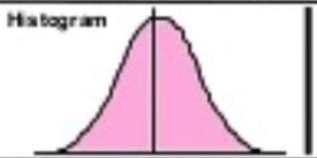
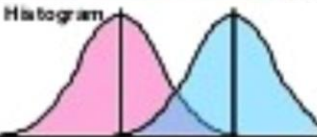
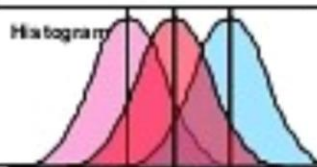
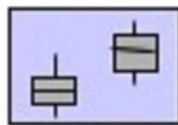
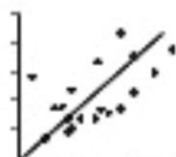
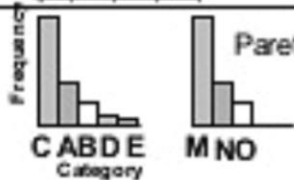

Hypothesis Testing with Python

USING STATISTICS FOR FINDING SIGNIFICANCE

Contributors ￣_(\ツ)_/_

Vivek Chuadhary | Chintan Chitroda | Manvendra Singh



What The Tool Tests	Statistical Test	Graphical Test
Mean of population data is different from an established target.	1-Sample t-test Stat > Basic Statistics > 1-Sample t	Histogram 
Mean of population 1 is different from mean of population 2.	2-Sample t-test Stat > Basic Statistics > 2-Sample t	Histogram 
The means of two or more populations is different.	1-Way ANOVA Stat > ANOVA > One-Way	Histogram 
Variance among two or more populations is different.	Homogeneity of Variance Stat > ANOVA > Homogeneity of Variance	 Box Plots
Output (Y) changes as the input (X) changes.	Linear Regression Stat > Regression > Fitted Line Plot	 Scatter Plots
Output counts from two two or more subgroups differ.	Chi-Square Test of Independence Stat > Tables > Cross Tabulation OR Chi-Square Test	 Pareto
Data is normally distributed	Normality Test Stat > Basic Statistics	

Hey Data Science Enthusiast, again we are back with one of the mini book on applied statistics with some of the methods & this is the basic version of the book, very soon another part will be out.

Everyone out there learn statistics but they always fail to apply when it comes to solving any project why?

Because out of 100% almost 80% to 85% are scared about Research, Statistics & applying your commensence.

When it comes to commensence you can not master with any course it will be develop by your curiosity & understanding the problem statement deeply that what makes you get into by taking this initial step.

Before applying statistics you have to think about assumption as per your given problem statement as statistics work on assumption but likelihood to be true, what it means, always statistics can't give you the true result & you have to compare particular result after applying statistics with your strong domain knowledge that you are working with (understanding problem statement deeply & strongly).

In this book we have cover some of the stuff that may help you out to explore & some of the resources have been taken from other parts as well to combine it in a good manner whether many of the stuff are custom.

We wish you will love this book & we are coming with update version for the same in 4 to 5 upcoming version.

Happy Learning

What is Hypothesis Testing and Why we do it ?

Hypothesis testing is a statistical method that is used in making statistical decisions using experimental data. Hypothesis Testing is basically an assumption that we make about the population parameter. -- Google

In simple words we make a Yes (Significant) or No (Not Significant) decision using Statistics using a sample of population data to check significance between features.

we have to make decisions about the hypothesis. These decisions include deciding if we should accept the null hypothesis or if we should reject the null hypothesis. Every test in hypothesis testing produces the significance value for that particular test. In Hypothesis testing, if the significance value of the test is greater than the predetermined significance level, then we accept the null hypothesis. If the significance value is less than the predetermined value, then we should reject the null hypothesis.

For example,

if we want to see the degree of relationship between two stock prices and the significance value of the correlation coefficient is greater than the predetermined significance level, then we can accept the null hypothesis and conclude that there was no relationship between the two stock prices. However, due to the chance factor, it shows a relationship between the variables.

Terms you should be familiar with

1. Null hypothesis:

- Null hypothesis is a statistical hypothesis that assumes that the observation is due to a chance factor. Null hypothesis is denoted by; $H_0: \mu_1 = \mu_2$, which shows that there is no difference between the two population means.

2. Alternative hypothesis:

- Contrary to the null hypothesis, the alternative hypothesis shows that observations are the result of a real effect.

3. Level of significance / P-value:

- Refers to the degree of significance in which we accept or reject the null-hypothesis. 100% accuracy is not possible for accepting or rejecting a hypothesis, so we therefore select a level of significance that is usually 5%.

4. Type I error:

- When we reject the null hypothesis, although that hypothesis was true. Type I error is denoted by alpha. In hypothesis testing, the normal curve that shows the critical region is called the alpha region.

5. Type II errors:

- When we accept the null hypothesis but it is false. Type II errors are denoted by beta. In Hypothesis testing, the normal curve that shows the acceptance region is called the beta region.

Types of Hypothesis Testing:

1. T-Test

- t-test is used to compare the mean of two given samples

2. Anova Test

- Anova Test is used to compare multiple (three or more) samples with a single test

3. Chi-Square Test

- Chi-square test is used to compare categorical variables

4. Pearson Correlation

- it is used to compare two numerical variables

Datasets used in this file can be found here:

https://drive.google.com/open?id=1u0YImKHCPahDReepW0Nru2f2RfDX_UjO

Let's start by look on simple examples using various testing methods

Hypothesis Testing , T-Testing

```
In [3]: #creating random data set of different weights for individuals  
average_weight = [33,34,35,36,32,28,29,30,31,37,36,35,33,34,31,40,24]
```

Hypothesis testing is all about assumption we create.

Average weight in class 12th is 35, Means Hypothesized mean is 35

Null Hypothesis , H_0 = Average age in class 12th is 35

Alt Hypothesis , H_a = Average age in class 12th is not 35

```
In [14]: from scipy import stats #importing stats package
```

One Sample T-Test

One Sample T-test is used for one parameter as seen in example we are looking to verify whether avarage age in class 12th is 35 or not

```
In [15]: stats.ttest_1samp(average_weight,35)  
  
Ttest_1sampResult(statistic=-2.354253623010381, pvalue=0.03166804359862131)
```

P value = 0.031 = 3.1%, This means that the probablity (or chance) of avaerage_weight 35 is only 3.1%. That is our Null Hypothesis is Wrong.

Generalizing, if P value < 5 % , we REJECT Null Hypothesis.

In our example, we REJECT H_0 , and conclude H_a that average_age in class 12th is NOT 35

Two sample Independent T-test

Used for comparing two parameters & to verify our assumptions.

```
In [16]: #creating an random data set of student in class 11th with each student weight  
average_weight1 = [29,31,28,33,31,34,32,20,32,28,27,26,30,31,34,30]
```

```
In [18]: average_weight #average weight of class 12th student as seen in One-Sample T-Test  
  
[33, 34, 35, 36, 32, 28, 29, 30, 31, 37, 36, 35, 33, 34, 31, 40, 24]
```

creating our assumptions

Null Hypothesis , H_0 = Avaerage_weight of class 12th & class 11th student is same.

Alt Hypothesis , H_a = Avaerage_weight of class 12th & class 11th student is not same.

```
In [20]: stats.ttest_ind(average_weight,average_weight1)
```

```
Ttest_indResult(statistic=2.404544177024533, pvalue=0.022355127034138323)
```

P value = 0.022 = 2.2%, This means that the probability (or chance) of average_weight of class 12th & class 11th students is same is only 2.2%. Null Hypothesis is Wrong.

We REJECT Null Hypothesis.

Concluding,Average_weight of class 12th & class 11th student is not same.

Two Sample paired (or Relational) T - test

We use Two Sample paired T-Test with keep in mind in simple way to check the effect before & after.

Studying effect of metaphor medicine on headache for individual who are suffering from migrain.

let's create random data set of individuals who are suffering from headache & we have given them two medicines one is paracetamol & another is metaphor,took the reading before having metaphor & after having metaphor.

```
In [21]: #Let's create random data set of individuals who are suffering from headache & we have given them two medic
before_metaphor = [68,45,46,34,23,67,80,120,34,54,68]
after_metaphor = [28,25,26,24,13,37,30,30,54,34,38]
```

H NULL = H0 = Response times before and after metaphor are same. This means Metaphor has NO EFFECT

H Alternative = Ha = Response times before and after Metaphor are NOT same. This means Metaphor has EFFECT

```
In [22]: stats.ttest_rel(before_metaphor,after_metaphor)
```

```
Ttest_relResult(statistic=3.2771720738937873, pvalue=0.00832867082029929)
```

P value = 0.008, 0.8% . $P < 5\%$, So we reject H0 and accept Ha, This means Metaphor has EFFECT on migrain suffered individuals

End of T-tests Tutorial

T-Test Example (Bike-Sharing Dataset)

We learn the theoratical concepts from abc institute but fail to implement in real world. Here we will see the use of hypothesis in Model Building.

What are we going to learn ?

- 1. Different types of hypothesis testing.
- 2. Their implications in model building.

T- Test

T-test is mostly used to check the difference in the means of two samples

Pre-processing

The first step always is to pre-process the dataset.

```
In [0]: #install researchpy
!pip install researchpy
## it combines pandas, scipy.stats and statsmodels to
##get more complete information in a single API call

Requirement already satisfied: researchpy in /usr/local/lib/python3.6/dist-packages (0.1.9)
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from researchpy) (1.18.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.6/dist-packages (from researchpy) (1.0.3)
Requirement already satisfied: scipy in /usr/local/lib/python3.6/dist-packages (from researchpy) (1.4.1)
Requirement already satisfied: statsmodels in /usr/local/lib/python3.6/dist-packages (from researchpy) (0.10.2)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.6/dist-packages (from pandas->researchpy) (2018.9)
Requirement already satisfied: python-dateutil>=2.6.1 in /usr/local/lib/python3.6/dist-packages (from pandas->researchpy) (2.8.1)
Requirement already satisfied: patsy>=0.4.0 in /usr/local/lib/python3.6/dist-packages (from statsmodels->researchpy) (0.5.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/dist-packages (from python-dateutil>=2.6.1->pandas->researchpy) (1.12.0)

In [0]: #import the libraries
import statsmodels.api as sm
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import scale
import researchpy as rc
import warnings
from scipy import stats
%matplotlib inline

In [0]: #read the data
df = pd.read_csv('/content/drive/My Drive/data_set/bike_sharing.csv')

In [0]: #check the shape
df.shape

(10886, 12)
```

```
In [0]: #check the head
df.head()
```

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	cou
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	32
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	10	13
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	1	1

```
In [0]: #check the information
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 12 columns):
#   Column      Non-Null Count  Dtype
---  -
0   datetime    10886 non-null  object
1   season      10886 non-null  int64
2   holiday     10886 non-null  int64
3   workingday  10886 non-null  int64
4   weather     10886 non-null  int64
5   temp        10886 non-null  float64
6   atemp       10886 non-null  float64
7   humidity    10886 non-null  int64
8   windspeed   10886 non-null  float64
9   casual      10886 non-null  int64
10  registered  10886 non-null  int64
11  count       10886 non-null  int64
dtypes: float64(3), int64(8), object(1)
memory usage: 1020.7+ KB
```

- 1. Most of the values are int64 and float64 types.
- 2. Only datetime feature is object type. And We can see there are no null values.

```
In [0]: #check the number of null values in each column
df.isnull().sum()
```

```
datetime    0
season      0
holiday     0
workingday  0
weather     0
temp        0
atemp       0
humidity    0
windspeed   0
casual      0
registered  0
count       0
dtype: int64
```

```
In [0]: df['atemp'].corr(df['temp']) #atemp and temp are correlated

0.9849481104817068
```

Atemp and temp has correlation of 0.985. They are providing the same information. We will drop the atemp feature and also datetime for our simplicity.

```
In [0]: #drop datetime
df.drop(['datetime','atemp'],axis = 1,inplace=True)
```

Now check the columns. Keep in mind that we have dropped datetime and atemp from our dataset.

```
In [0]: #check the unique values in each column
df.apply(lambda x : x.nunique())
```

```
season      4
holiday     2
workingday  2
weather     4
temp       49
humidity    89
windspeed   28
casual     309
registered  731
count      822
dtype: int64
```

```
In [0]: #standardize all the numerical features
num_scaled = scale (df[['temp','humidity','windspeed','casual','registered']],copy=False)
#scale takes the difference of each values from the mean and divide by standard deviation
num_scaled
```

```
array([[ -1.33366069,  0.99321305, -1.56775367, -0.66099193, -0.94385353],
       [-1.43890721,  0.94124921, -1.56775367, -0.56090822, -0.81805246],
       [-1.43890721,  0.94124921, -1.56775367, -0.62095844, -0.851158  ],
       ...,
       [-0.80742813, -0.04606385,  0.26970368, -0.64097518,  0.05593396],
       [-0.80742813, -0.04606385, -0.83244247, -0.48084125, -0.25525818],
       [-0.91267464,  0.21375537, -0.46560752, -0.64097518, -0.47375478]])
```

###Preparing Data for t-test

Now we will perform t-test to check whether the number of bike rentals are dependent on workingday or not. For this we will use two sample t-test.

Two sample t-test is used to check whether the means of two samples(group) are same or different. We want to check whether the number of bikes rented on working day are different then number of bikes rented on non-working days.

Let's check the mean of bikes rented on working and non-working days.

```
In [0]: df.groupby('workingday')['count'].describe()
```

	count	mean	std	min	25%	50%	75%	max
workingday								
0	3474.0	188.506621	173.724015	1.0	44.0	128.0	304.0	783.0
1	7412.0	193.011873	184.513659	1.0	41.0	151.0	277.0	977.0

We can see that mean on working days is 193.0 and mean on the non-working day is 188.5. Definitely we can see that there is difference in the means of working and non working days.

But the quetsion is, is this difference in the mean stastically significant or was it just due to random chance ?

Steps for performing hypothesis testing.

- 1. set up Null Hypothesis (H0)
- 2. State the alternate hypothesis (H1)
- 3. Set a significance level (alpha)
- 4. Calculate test Statistics.
- 5. Decision to accept or reject null hypothesis.

```
In [0]: #create 2 samples one for working days and one for non-working days
sample_01 = df[df['workingday'] == 1]
sample_02 = df[df['workingday'] == 0]
```

```
In [0]: #check the shape of both the samples
print(sample_01.shape,sample_02.shape)
```

```
(7412, 10) (3474, 10)
```

sample_01 have 7412 observations whereas sample_02 only have 3474 obsrvations. We have to take equal

number of observations in both the sample.

```
In [0]: #make equal number of records in each sample
sample_01 = sample_01.sample(3474)
print(sample_01.shape, sample_02.shape)

(3474, 10) (3474, 10)
```

Before directly jumping for hypothesis testing we have to check for different assumptions related to the kind of hypothesis test we want to perform.

##Assumption for T-Test

1. The variances of the 2 samples are equal(We will use Levene's test to check this assumption).
2. The distrubtion of the residuals b/w the two groups should follow the normal distribution. We can plot histogram and see whether the distribution follows the normal distribution or not. We can also plot a Q-Q plot. We can check the normality using shapiro-wilks test as well.

```
In [0]: #Levene's test to check whether the variances of the two group are same.
#H0 : Variances are same.
#H1 : Variances are not same.
#Alpha = 0.05%
#if p-value > alpha (Cannot reject H0)
#if p-value < alpha (Accept null hypothesis)
```

```
In [0]: alpha = 0.05
Stats, Pvalue = stats.levene(sample_01['count'], sample_01['count'])
print(f' Test statistics : {Stats} \n Alpha : {alpha} \n P-value : {Pvalue}')
if Pvalue > alpha:
    print(' Variances are same accept null hypothesis ')
else:
    print(' Variances are not same reject not null hypothesis ')

Test statistics : 0.0
Alpha : 0.05
P-value : 1.0
Variances are same accept null hypothesis
```

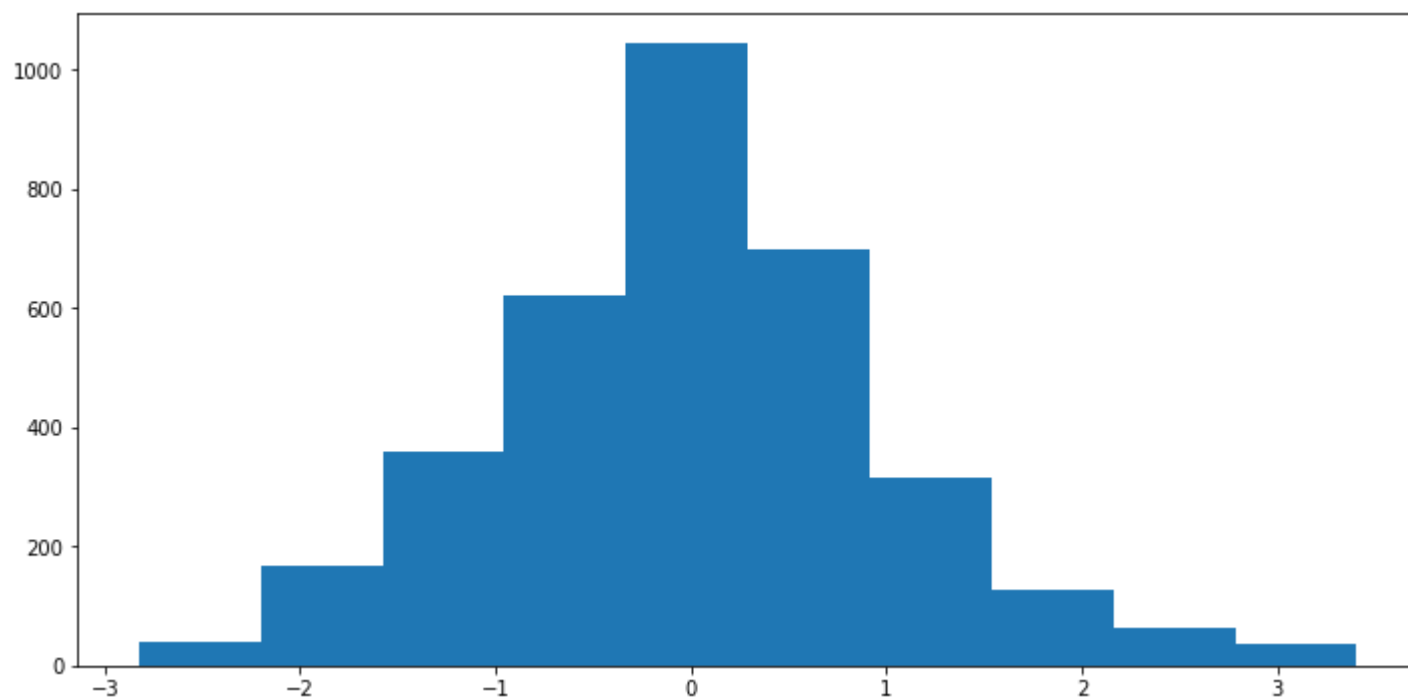
Here we have got 2 things:

1. Test Statistics
2. And p-value associated with test stastics. We can see that p-value(1.0) > alpha(0.05). So we fail to reject the null hypothesis. Variances of the 2 samples are equal.

###Check for normality

Take the difference between two samples and scale it to check the normality of the residuals.

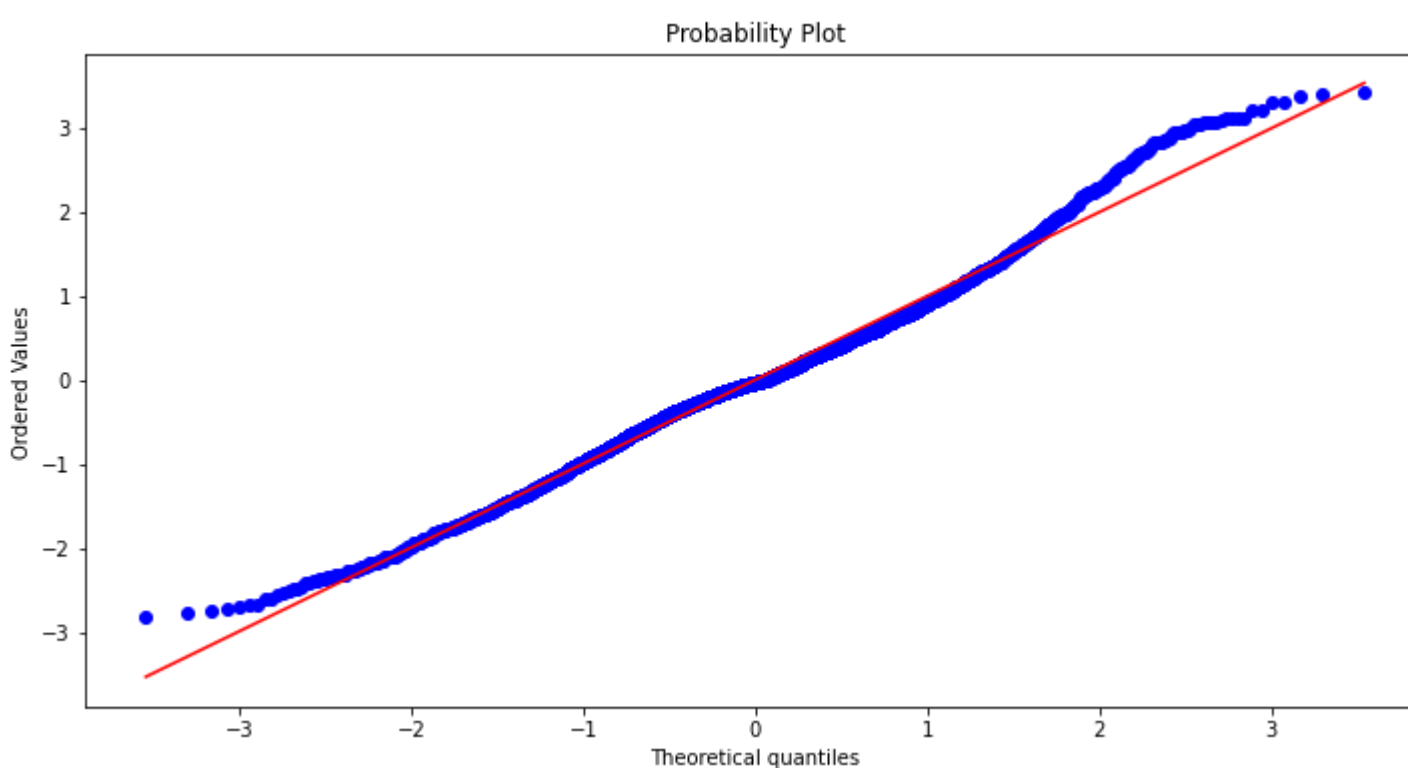
```
In [0]: #we will take the difference b/w sample_01 and sample_02 and plot a histogram to check for normality
#we will scale the difference
diff = scale((np.array(sample_01['count']) - np.array(sample_02['count'])))
plt.figure(figsize=(12,6))
plt.hist(diff)
plt.show()
```



The distribution seems very close to normal distribution. Let's check other methods to check the normality of the residuals.

Q-Q plot, Generates the a probability of sample data against the quantiles of theoretical distributions.

```
In [0]: #q-q plot to check the normality
plt.figure(figsize=(12,6))
stats.probplot(diff,plot=plt,dist='norm')
plt.show()
```



When the points are closely follows the redline we can say that the residulas are normally distributed. Here we see that after 2 standard deviation the points are scattered from redline. They doesn't follow the redline. But most of the data points are still close to the redline so we accept the assumption of normality.

Till now we have seen graphical methods to represent to check the assumption of normality. Now let's check is it with statstical test (Shapiro-Wilk Test)

```
In [0]: #Stastical test for checking normality
#Shapiro-wilk test
#H0 : Normally distributed
#H1 : Not Normally distributed
```

```
In [0]: alpha = 0.05
statistic,p_value = stats.shapiro(diff)
if p_value > alpha:
    print(f'Accept Null Hypothesis p-value : {p_value}')
else:
    print(f'Reject Null Hypothesis p-value : {p_value}')
```

Reject Null Hypothesis p-value : 2.197166195727976e-14

Here shapiro willk test shows that the residuals are not normally distributed. for demonstration purpose We will continue with t-test, but in practice we should not perform t-test when the assumption of normality is violated.

##Independent Sample T-test

```
In [0]: # H0 : There's no difference in mean (Bike rental doesn't depends on workingday)
# H1 : There's a difference in mean (Bike rental depends on workingday)
# Alpha : 0.05%
alpha = 0.05
statistic , p_value = stats.ttest_ind(sample_01['count'],sample_02['count'])
if p_value > alpha:
    print(f'Fail to reject Null Hypothesis p-value is {p_value}')
else:
    print('Reject Null Hypothesis')
```

Fail to reject Null Hypothesis p-value is 0.06442712309437282

As we can see that the p-value is greater than alpha. So we can't reject our null hypothesis.

working day has no effect on number of bikes rented.

Comparing means, Student's t-test, ANOVA

In this Notebook, we will examine the concept of comparing means between groups of data using Student's t-test and ANOVA.

```
In [1]: import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as st
import seaborn as sns
```

executed in 1.23s, finished 17:06:37 2020-04-22

Let's Say any company 'ABC' class I.Q. test

ABC company boss at a certain MNC claims that the employee in his company are above average intelligence. A random sample of 30 employee have a mean score of 112.5. **Is there sufficient evidence to support an compay Boss claim?** The mean population IQ is 100 with a standard deviation of 15.

Note, although this problem does not sound like an example of comparing group means, it is indeed the same. Instead of comparing two sample datasets, we are comparing one sample dataset to the population mean.

Some additional questions can also be asked to understand the impact of various quantities involved,

- What happens to our conclusion if the population mean is higher, say 107?
- What happens to our conclusion if the population standard deviation is lower, say 10? What happens if it is higher instead?
- What happens if there were only 4 employee who could be tested for the I.Q.? What if we could test 100 employee instead?
- What is the impact of changing the significance level to 0.01 from 0.05?

Given data

```
In [2]: population_mean = 100
population_std = 15
n_sample = 30
```

executed in 3ms, finished 17:06:37 2020-04-22

Generate typical distribution of a Employee at company 'ABC'

This will be considered an 'average' class i.e. not having above-average intelligence as the MNC Boss claims.

```
In [3]: avg_class = np.vectorize(int)(np.random.normal(loc=population_mean,scale=population_std,size=n_sample))
```

executed in 104ms, finished 17:06:37 2020-04-22

```
In [4]: print("A typical class I.Q.:",avg_class)
```

executed in 11ms, finished 17:06:37 2020-04-22

A typical class I.Q.: [77 93 98 89 90 99 105 117 102 92 76 76 110 98 132 99 85 110
 94 94 76 85 106 96 119 111 111 106 96 93]

The given class data (generated with the given mean and assumed same variance as population)

```
In [5]: given_class = np.vectorize(int)(np.random.normal(loc=112.5,scale=population_std,size=n_sample))
```

executed in 4ms, finished 17:06:38 2020-04-22

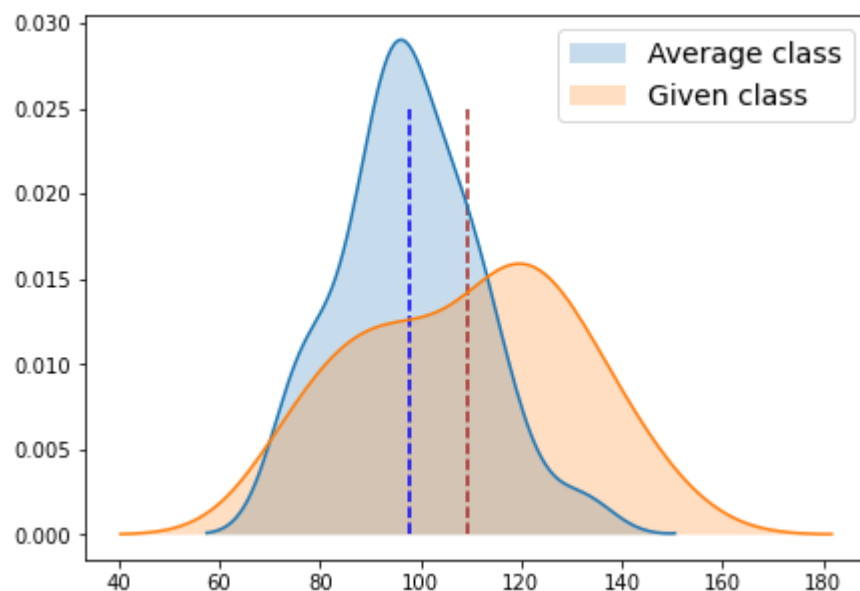
```
In [6]: print("Given class I.Q.:",given_class)
```

executed in 5ms, finished 17:06:38 2020-04-22

Given class I.Q.: [128 115 132 80 96 95 142 122 93 114 95 77 110 148 123 108 82 119
128 74 97 77 122 137 119 119 89 93 115 134]

```
In [7]: plt.figure(figsize=(7,5))
sns.kdeplot(avg_class,shade=True)
sns.kdeplot(given_class,shade=True)
plt.legend(['Average class','Given class'],fontsize=14)
plt.vlines(x=avg_class.mean(),ymin=0,ymax=0.025,color='blue',linestyle='--')
plt.vlines(x=given_class.mean(),ymin=0,ymax=0.025,color='brown',linestyle='--')
plt.show()
```

executed in 230ms, finished 17:06:38 2020-04-22



```
In [8]: std_err = population_std/np.sqrt(n_sample)
z_stat = (given_class.mean()-population_mean)/std_err

print("Standard error of the mean:",std_err)
print("Z-statistic:",z_stat)
```

executed in 4ms, finished 17:06:38 2020-04-22

```
Standard error of the mean: 2.7386127875258306
Z-statistic: 3.444566306088046
```

```
In [9]: alpha = 0.05
rejection_threshold = st.norm.ppf(1-alpha)
```

executed in 4ms, finished 17:06:39 2020-04-22

```
In [10]: if z_stat>rejection_threshold:
    print("We reject the NULL hypothesis. The class I.Q. is indeed above average")
else:
    print("We cannot reject the NULL hypothesis that class average is same as population average.")
```

executed in 4ms, finished 17:06:41 2020-04-22

```
We reject the NULL hypothesis. The class I.Q. is indeed above average
```



```
In [11]: def hypothesis_testing(n_sample=30,population_mean=100,population_std=15,alpha=0.05):
        """
        Tests the hypothesis of above average I.Q. and reports the conclusion
        """
        given_class=np.vectorize(int)(np.random.normal(loc=112.5,scale=population_std,size=n_sample))

        std_err = population_std/np.sqrt(n_sample)
        z_stat = (given_class.mean()-population_mean)/std_err

        alpha = 0.05
        rejection_threshold = st.norm.ppf(1-alpha)

        if z_stat>rejection_threshold:
            print("We reject the NULL hypothesis. The class I.Q. is indeed above average")
        else:
            print("We cannot reject the NULL hypothesis that class average is same as population average.")
```

executed in 6ms, finished 17:06:45 2020-04-22

Test with default values

```
In [12]: hypothesis_testing()
```

executed in 3ms, finished 17:06:45 2020-04-22

We reject the NULL hypothesis. The class I.Q. is indeed above average

What if the population mean is higher 110?

```
In [13]: hypothesis_testing(population_mean=107)
```

executed in 4ms, finished 17:06:46 2020-04-22

We cannot reject the NULL hypothesis that class average is same as population average.

What if the population standard deviation is lower, say 10? What happens if it is higher instead?

```
In [14]: hypothesis_testing(population_std=10)
```

executed in 4ms, finished 17:06:47 2020-04-22

We reject the NULL hypothesis. The class I.Q. is indeed above average

```
In [15]: hypothesis_testing(population_std=40)
```

executed in 4ms, finished 17:06:47 2020-04-22

We cannot reject the NULL hypothesis that class average is same as population average.

What if there were only 4 employee from ABC company? What if we could test 100 employee instead?

```
In [16]: hypothesis_testing(n_sample=4)
```

executed in 3ms, finished 17:06:47 2020-04-22

We reject the NULL hypothesis. The class I.Q. is indeed above average

```
In [17]: hypothesis_testing(n_sample=100)
```

executed in 4ms, finished 17:06:48 2020-04-22

We reject the NULL hypothesis. The class I.Q. is indeed above average

What is the impact of changing the significance level to 0.01 (or even 0.001) from 0.05?

```
In [18]: hypothesis_testing(alpha=0.01)

executed in 4ms, finished 17:06:48 2020-04-22

We reject the NULL hypothesis. The class I.Q. is indeed above average
```

```
In [19]: hypothesis_testing(alpha=0.001)

executed in 4ms, finished 17:06:49 2020-04-22

We reject the NULL hypothesis. The class I.Q. is indeed above average
```

Independent Student's t-test implementation

The Student’s t-Test is a statistical hypothesis test for testing whether two samples are expected to have been drawn from the same population.

It is named for the pseudonym “Student” used by [William Gosset](https://en.wikipedia.org/wiki/William_Sealy_Gosset) (https://en.wikipedia.org/wiki/William_Sealy_Gosset), who developed the test.

The test works by checking the means from two samples to see if they are significantly different from each other. It does this by calculating the standard error in the difference between means, which can be interpreted to see how likely the difference is, if the two samples have the same mean (the null hypothesis).

The t-statistic calculated by the test can be interpreted by comparing it to critical values from the [t-distribution](https://en.wikipedia.org/wiki/Student%27s_t-distribution) (https://en.wikipedia.org/wiki/Student%27s_t-distribution). The critical value can be calculated using the degrees of freedom and a significance level with the percent point function (PPF).

We can interpret the statistic value in a two-tailed test, meaning that if we reject the null hypothesis, it could be because the first mean is smaller or greater than the second mean.

A custom function

```
In [20]: def independent_ttest(data1, data2, alpha=0.05):
        """
        Student's t-test for independent groups

        Argument:
            data1: First group data in numpy array format
            data2: Second group two data in numpy array format
            alpha: Significance level

        Returns:
            t_stat: Computed t-statistic
            df: Degrees of freedom
            cv: Critical value
            p: p-value (of NULL hypothesis)
        """

        import scipy.stats as st
        # calculate means
        mean1, mean2 = np.mean(data1), np.mean(data2)
        # calculate standard errors
        se1, se2 = st.sem(data1), st.sem(data2)
        # standard error on the difference between the samples
        sed = np.sqrt(se1**2.0 + se2**2.0)
        # calculate the t statistic
        t_stat = (mean1 - mean2) / sed
        # degrees of freedom
        df = len(data1) + len(data2) - 2
        # calculate the critical value
        cv = st.t.ppf(1.0 - alpha, df)
        # calculate the p-value
        p = (1.0 - st.t.cdf(abs(t_stat), df)) * 2.0
        # return everything
        return t_stat, df, cv, p

executed in 8ms, finished 17:06:50 2020-04-22
```

Apply the function for testing hypothesis of equal mean

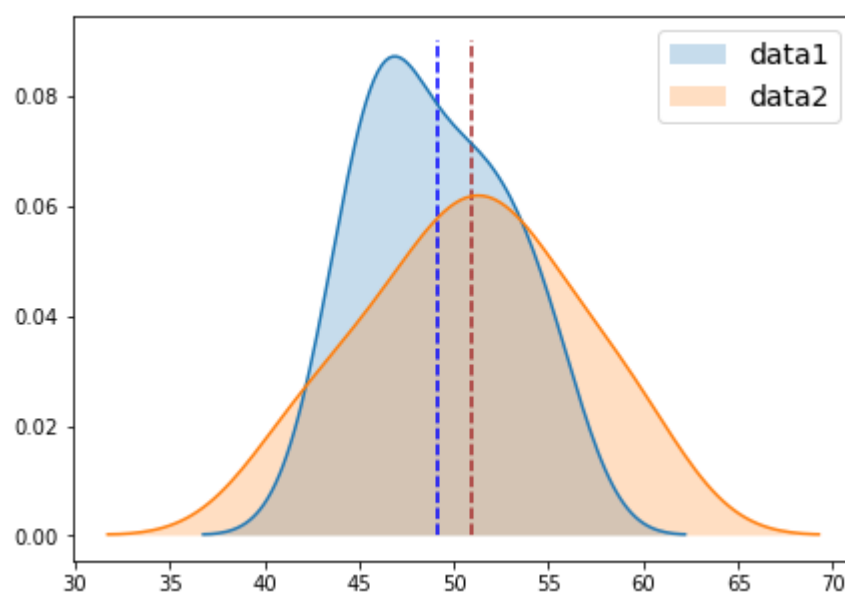
```
In [21]: n_sample = 20
```

executed in 2ms, finished 17:06:50 2020-04-22

```
In [22]: data1 = 5 * np.random.randn(n_sample) + 50
data2 = 5 * np.random.randn(n_sample) + 51

plt.figure(figsize=(7,5))
sns.kdeplot(data1,shade=True)
sns.kdeplot(data2,shade=True)
plt.legend(['data1', 'data2'],fontsize=14)
plt.vlines(x=data1.mean(),ymin=0,ymax=0.09,color='blue',linestyle='--')
plt.vlines(x=data2.mean(),ymin=0,ymax=0.09,color='brown',linestyle='--')
plt.show()
```

executed in 189ms, finished 17:06:51 2020-04-22



```
In [23]: # calculate the t test
alpha = 0.05
t_stat, df, cv, p = independent_ttest(data1, data2, alpha)
print('t=%.3f, df=%d, cv=%.3f, p=%.3f' % (t_stat, df, cv, p))
print()

# interpret via critical value
if abs(t_stat) <= cv:
    print('Fail to reject null hypothesis that the means are equal.')
else:
    print('Reject the null hypothesis that the means are equal.')
# interpret via p-value
if p > alpha:
    print('Fail to reject null hypothesis that the means are equal.')
else:
    print('Reject the null hypothesis that the means are equal.')
```

executed in 36ms, finished 17:06:51 2020-04-22

t=-1.232, df=38, cv=1.686, p=0.226

Fail to reject null hypothesis that the means are equal.
Fail to reject null hypothesis that the means are equal.

More numner of sample, more _statistical power_!

The power of a binary hypothesis test is the probability that the test rejects the null hypothesis (H_0) when a specific alternative hypothesis (H_a) is true.

The statistical power ranges from 0 to 1, and **as statistical power increases, the probability of making a type II error (wrongly failing to reject the null hypothesis) decreases**. For a type II error probability of β , the corresponding statistical power is $1 - \beta$.

For example, if experiment 1 has a statistical power of 0.7, and experiment 2 has a statistical power of 0.95, then there is a stronger probability that experiment 1 had a type II error than experiment 2, and experiment 2 is more reliable than experiment 1 due to the reduction in probability of a type II error.

It can be equivalently thought of as the probability of accepting the alternative hypothesis (H_a) when it is true—that is, the **ability of a test to detect a specific effect**, if that specific effect actually exists.

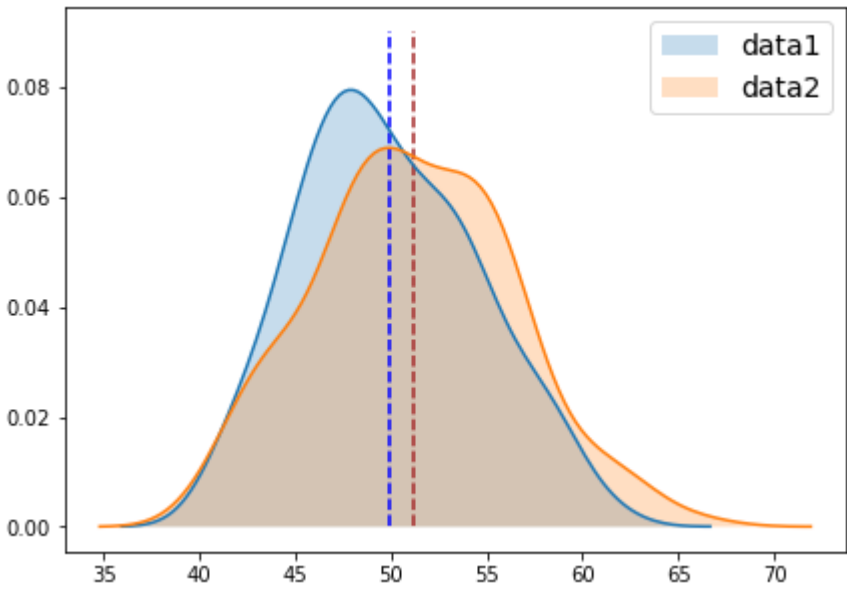
```
In [24]: n_sample = 200
```

executed in 3ms, finished 17:06:52 2020-04-22

```
In [25]: data1 = 5 * np.random.randn(n_sample) + 50
data2 = 5 * np.random.randn(n_sample) + 51

plt.figure(figsize=(7,5))
sns.kdeplot(data1,shade=True)
sns.kdeplot(data2,shade=True)
plt.legend(['data1','data2'],fontsize=14)
plt.vlines(x=data1.mean(),ymin=0,ymax=0.09,color='blue',linestyle='--')
plt.vlines(x=data2.mean(),ymin=0,ymax=0.09,color='brown',linestyle='--')
plt.show()
```

executed in 187ms, finished 17:06:52 2020-04-22



```
In [26]: # calculate the t test
alpha = 0.05
t_stat, df, cv, p = independent_ttest(data1, data2, alpha)
print('t=%.3f, df=%d, cv=%.3f, p=%.3f' % (t_stat, df, cv, p))
print()

# interpret via critical value
if abs(t_stat) <= cv:
    print('Fail to reject null hypothesis that the means are equal.')
else:
    print('Reject the null hypothesis that the means are equal.')
# interpret via p-value
if p > alpha:
    print('Fail to reject null hypothesis that the means are equal.')
else:
    print('Reject the null hypothesis that the means are equal.')
```

executed in 6ms, finished 17:06:52 2020-04-22

```
t=-2.571, df=398, cv=1.649, p=0.010

Reject the null hypothesis that the means are equal.
Reject the null hypothesis that the means are equal.
```

ANOVA using Scipy (`f_oneway()` method)

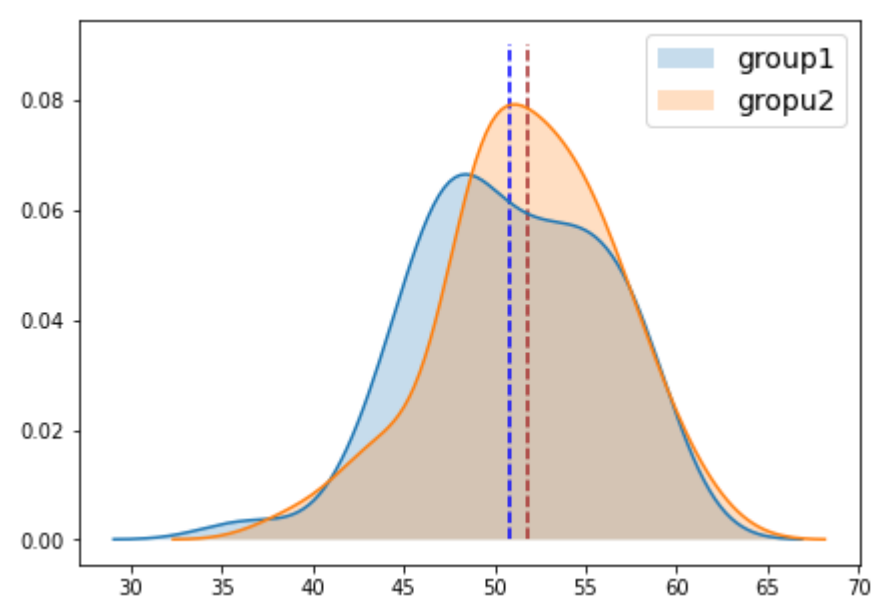
```
In [27]: n_sample = 50
```

executed in 3ms, finished 17:06:53 2020-04-22

```
In [28]: group1 = 5 * np.random.randn(n_sample) + 50
group2 = 5 * np.random.randn(n_sample) + 52

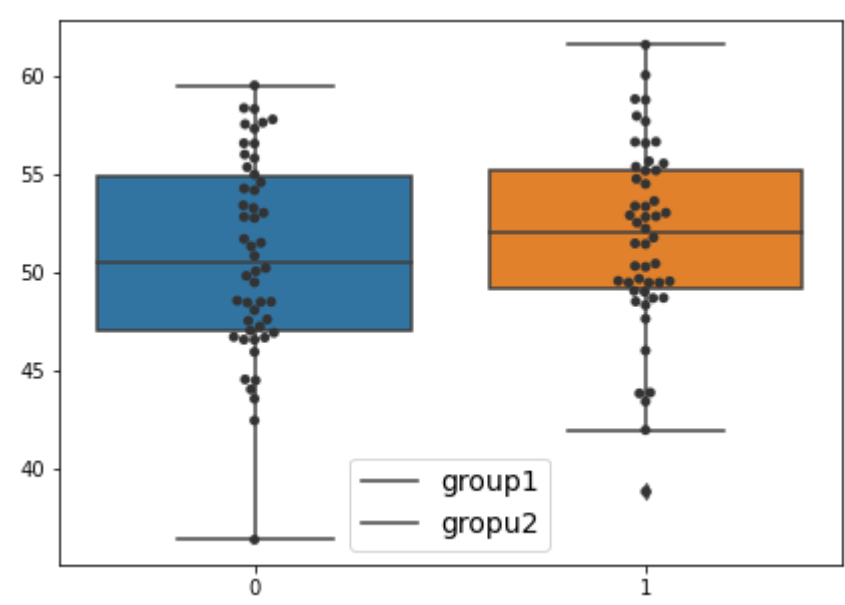
plt.figure(figsize=(7,5))
sns.kdeplot(group1,shade=True)
sns.kdeplot(group2,shade=True)
plt.legend(['group1', 'gropu2'],fontsize=14)
plt.vlines(x=group1.mean(),ymin=0,ymax=0.09,color='blue',linestyle='--')
plt.vlines(x=group2.mean(),ymin=0,ymax=0.09,color='brown',linestyle='--')
plt.show()
```

executed in 190ms, finished 17:06:53 2020-04-22



```
In [29]: plt.figure(figsize=(7,5))
sns.boxplot(data=[group1,group2])
sns.swarmplot(data=[group1,group2],color='.2')
plt.legend(['group1', 'gropu2'],fontsize=14)
plt.show()
```

executed in 185ms, finished 17:06:53 2020-04-22



```
In [30]: f,p=st.f_oneway(group1,group2)
```

executed in 3ms, finished 17:06:54 2020-04-22


```
In [31]: print("The F-statistic obtained running ANOVA on the data groups:",f)
print("The p-value of the ANOVA test:",p)
if p>0.05:
    print("\nANOVA fails to reject the hypothesis of equal mean")
else:
    print("\nWe reject the hypothesis of equal mean as per ANOVA test result")
```

executed in 4ms, finished 17:06:54 2020-04-22

The F-statistic obtained running ANOVA on the data groups: 0.9262824297071112
The p-value of the ANOVA test: 0.33819957501665976

ANOVA fails to reject the hypothesis of equal mean

For multiple groups, two-way pair comparisons are done

```
In [32]: def multi_anova(groups,alpha=0.05):
        """
        Two-way ANOVA between multiple groups
        groups: A dictionary object of trial groups
        """

        from itertools import combinations
        list_anova = list(combinations(list(groups.keys()),2))

        for comb in list_anova:
            _,p=st.f_oneway(groups[comb[0]],groups[comb[1]])
            if p>0.05:
                print("\nANOVA fails to reject the hypothesis of equal mean for {} and {}".format(comb[0],comb[1]))
            else:
                print("\nWe reject the hypothesis of equal mean for {} and {} as per ANOVA test result".format(comb[0],comb[1]))
```

executed in 6ms, finished 17:06:54 2020-04-22

Imaginary drug trial

Let's suppose we have three versions of a blood pressure medicine developed - 'A','B','C'.

We give this medicine to four groups - three trial groups and one control group, whose participants were given placebo (being a blind experiment, they did not know). There were total 155 participants and due to some logistic reason, the divisions were slightly uneven.

No problem, ANOVA can handle slight difference in the sample count among groups!

As a data scientist, you have been asked to analyze the clinical trial data and recommend whether to go ahead with further development on any (or more than one) of these drugs.

Since they all the variations of same fundamental molecule, their impact is not drastically different and depending on trial participants physiology, there are lot of variations among the trial groups.

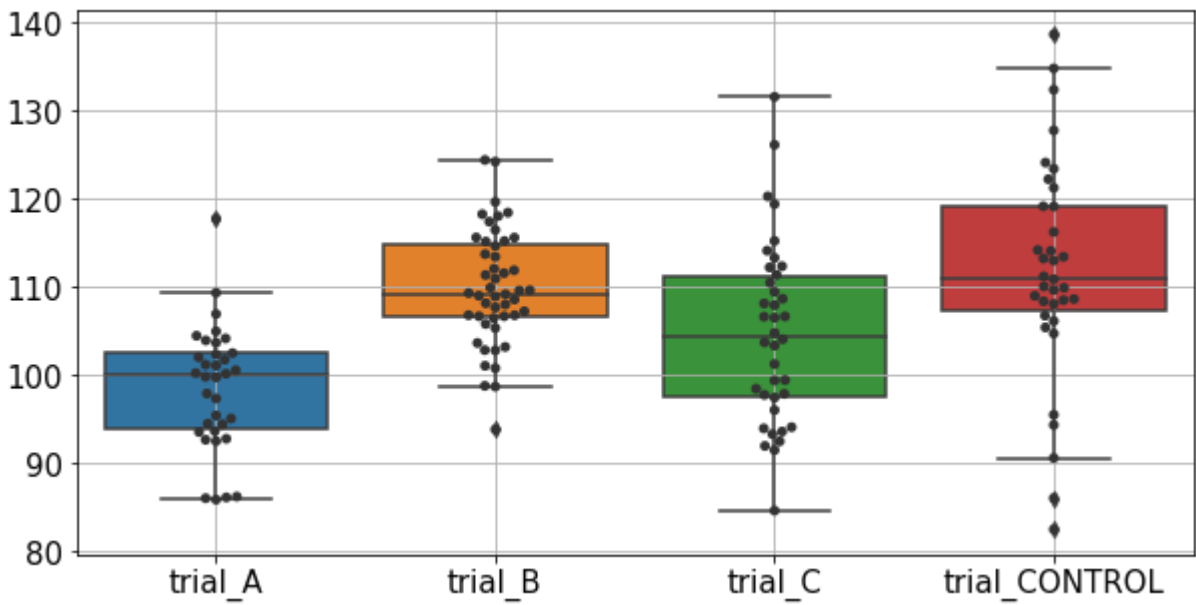
Let's put our `multi_anova` function to use for this imaginary study!

```
In [33]: trial_A = 9*np.random.randn(34) + 102
trial_B = 6*np.random.randn(48) + 109
trial_C = 12*np.random.randn(38) + 103
trial_CONTROL = 10*np.random.randn(35) + 110
```

executed in 5ms, finished 17:06:55 2020-04-22

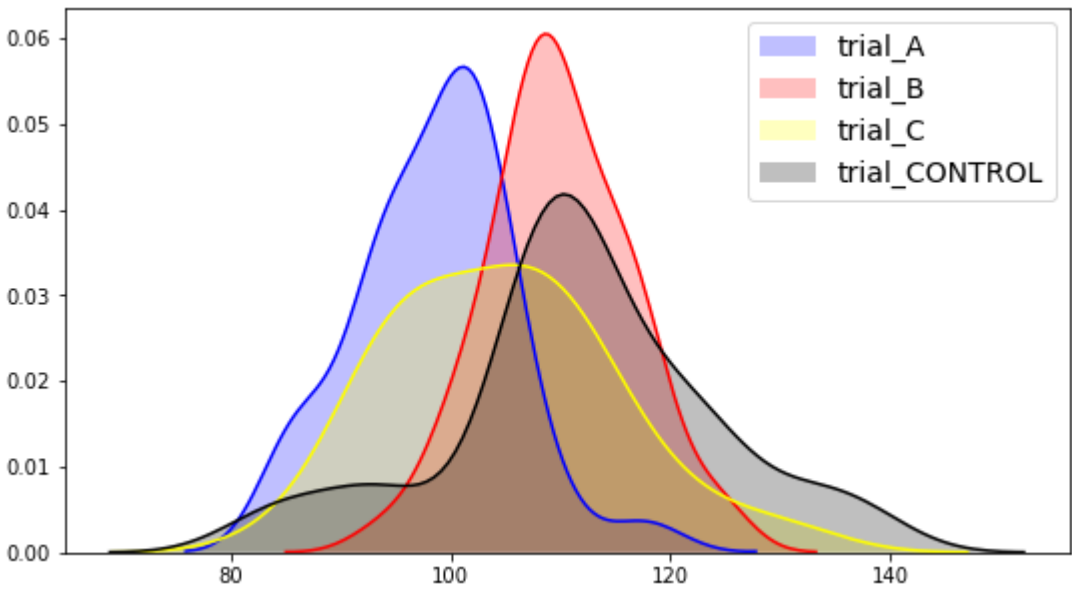
```
In [34]: plt.figure(figsize=(10,5))
ax=sns.boxplot(data=[trial_A,trial_B,trial_C,trial_CONTROL])
ax=sns.swarmplot(data=[trial_A,trial_B,trial_C,trial_CONTROL],color='.2')
ax.set_xticklabels(['trial_A','trial_B','trial_C','trial_CONTROL'],fontsize=15)
ax.tick_params(axis="y", labels=15)
plt.grid(True)
plt.show()
```

executed in 227ms, finished 17:06:55 2020-04-22



```
In [35]: plt.figure(figsize=(9,5))
sns.kdeplot(trial_A,shade=True,color='Blue')
sns.kdeplot(trial_B,shade=True,color='red')
sns.kdeplot(trial_C,shade=True,color='yellow')
sns.kdeplot(trial_CONTROL,shade=True,color='black')
plt.legend(['trial_A','trial_B','trial_C','trial_CONTROL'],fontsize=14)
plt.show()
```

executed in 244ms, finished 17:06:55 2020-04-22



```
In [36]: groups = {'trial_A':trial_A,'trial_B':trial_B,'trial_C':trial_C,'trial_CONTROL':trial_CONTROL}
```

executed in 3ms, finished 17:06:55 2020-04-22

```
In [37]: multi_anova(groups)
```

executed in 4ms, finished 17:06:56 2020-04-22

We reject the hypothesis of equal mean for trial_A and trial_B as per ANOVA test result

We reject the hypothesis of equal mean for trial_A and trial_C as per ANOVA test result

We reject the hypothesis of equal mean for trial_A and trial_CONTROL as per ANOVA test result

We reject the hypothesis of equal mean for trial_B and trial_C as per ANOVA test result

ANOVA fails to reject the hypothesis of equal mean for trial_B and trial_CONTROL

We reject the hypothesis of equal mean for trial_C and trial_CONTROL as per ANOVA test result

What's the conclusion?

From the results, printed out, we see that equal mean hypothesis (with CONTROL group) could not be rejected for trial_B, whereas the hypothesis was rejected for the cases - trial_A and trial_CONTROL, trial_B and trial_CONTROL.

Therefore, the trials of medicine A and medicine C showed statistically significant lowering of blood pressure whereas medicine B did not.

Let's understand ANOVA(Analysis of Variance)

You may know it's an Analysis of Variance but how to use in actual manner lot of individual are unaware & don't use such concepts while dealing with Data Set for any required problem statement

Let's take an example for the delivery time of different courier companies ,where A,B & C stands for three different companies like DTDC,FEDX & other. So,it's obvious that customer will send courier through the company which takes less time of delivery.

```
In [1]: import pandas as pd

In [2]: A = [12.6, 12, 11.8, 11.9, 13, 12.5, 14]
        B = [10, 10.2, 10, 12, 14, 13]
        C = [10.1, 13, 13.4, 12.9, 8.9, 10.7, 13.6, 12]

In [3]: all_scores = A + B + C
        company_names = (['A'] * len(A)) + (['B'] * len(B)) + (['C'] * len(C))

In [4]: data = pd.DataFrame({'company': company_names, 'score': all_scores})

In [5]: data.head(20)
```

	company	score
0	A	12.6
1	A	12.0
2	A	11.8
3	A	11.9
4	A	13.0
5	A	12.5
6	A	14.0
7	B	10.0
8	B	10.2
9	B	10.0
10	B	12.0
11	B	14.0
12	B	13.0
13	C	10.1
14	C	13.0
15	C	13.4
16	C	12.9
17	C	8.9
18	C	10.7
19	C	13.6

```
In [6]: data.groupby('company').mean() # here by calculate mean we can see avareage time to delivery the p
ackage & less delivery time
# can be seen with company B
```

score	
company	
A	12.542857
B	11.533333
C	11.825000

so,the delivery time per company which is faster with B

Did you think finding mean & saying that company B is faster in terms of delivering particular package??

No,as you can observed from the original Data that company B use to deliver the package at very slow rate as well.

Give a think on "Why ANOVA is known to be as "Analysis Of Variance""

It's known to be as "Analysis Of Variance" because ANOVA will estimate the total hours of variation in delivery time that different company use to delivery.

The ANOVA model starts by estimating the total amount of variation that exists in the delivery times(wikipedia)

```
In [7]: data.head(1)
```

company		score
0	A	12.6

```
In [8]: import seaborn as sns
sns.boxplot( x=data["company"], y=data["score"] )
```

<matplotlib.axes._subplots.AxesSubplot at 0x29edd2c108>

As we can see to our sample Data Set we can say that delivery time for the courier deliery ranges from 8,9 minuter to 14 minutes.

So,if new courier service want to give to their user estimate delivery time we can say it can be in between 8,9 & 14 minutes & let's call this total variation as of now.

The next step is to split this total variation in two: Between-Group Variation and Within-Group Variation

Between-Group Variation is Explained by our Variable Company

If we add the variable company in the graph, we see that if we know which company delivers our package, we can give a more precise range of delivery times.

If company A delivers, it takes between 11,8 and 14 minutes. If company B delivers, it takes between 10,0 and 14,0 minutes. If company C delivers, it takes between 8,9 and 13,6 minutes.

This phenomenon is due to the Between-Group variation: a quantification of the variation explained by our variable.

Within-Group Variation is not Explained by our Variable Company

However, there is also some part of variation that cannot be explained by our variable ‘Company’: we still don’t know why there is a difference between 11,8 and 14 in company A’s delivery times and we would need more variables to explain this. Since we do not have those new variables, the variation remains unexplained and is called the within-group variation.

ANOVA: hypothesis test for group differences

When the total variation is split in two, a hypothesis test is applied to find out whether the observed differences in our sample of 21 is significant:

Is one pizza company systematically faster, or is this random noise due to the sampling effect?

We need a statistical test to give us this answer: the ANOVA F-test.

ANOVA using statsmodels

```
In [9]: import statsmodels.api as sm
        from statsmodels.formula.api import ols

In [10]: lm = ols('score ~ company',data=data).fit()
         table = sm.stats.anova_lm(lm)
         print(table)
```

	df	sum_sq	mean_sq	F	PR(>F)
company	2.0	3.606905	1.803452	0.821297	0.455683
Residual	18.0	39.525476	2.195860	NaN	NaN

Sum of Squares Total

Let’s get to the action! What I described before as variation is mathematically measured by the Sum of Squares,

Sum of Squares To get there, we first compute the overall mean

```
In [11]: # compute overall mean
         overall_mean = data['score'].mean()
         overall_mean

11.980952380952381
```

And then compute the sum of the squared differences between the original scores and the overall mean:


```
In [12]: # compute Sum of Squares Total
data['overall_mean'] = overall_mean
ss_total = sum((data['score'] - data['overall_mean'])**2)
ss_total
```

43.132380952380956

This value can be found in the ANOVA table of statsmodels by taking the sum of the sum_sq column.

Sum of Squares Residual

The computation for residual Sum of Squares is slightly different because it takes not the overall average, but the three group averages.

We need to subtract each value from the mean of its group (the mean of its own courier company) and then square those differences and sum them. Here’s how it’s done in Python:

```
In [13]: # compute group means
group_means = data.groupby('company').mean()
group_means = group_means.rename(columns = {'score': 'group_mean'})
group_means
```

	group_mean	overall_mean
company		
A	12.542857	11.980952
B	11.533333	11.980952
C	11.825000	11.980952

```
In [14]: # add group means and overall mean to the original data frame
data = data.merge(group_means, left_on = 'company', right_index = True)
```

```
In [15]: # compute Sum of Squares Residual
ss_residual = sum((data['score'] - data['group_mean'])**2)
ss_residual
```

39.52547619047619

We can find this value in the ANOVA table of statsmodels under sum_sq at the line Residual.

Sum of Squares Explained

Having computed the total sum of squares and the residual sum of squares, we can now compute the Explained Sum of Squares using:

Summation of Sums of Squares Since we already have the SS-Residual and SS-Total, we could do a simple subtraction to get SS-Explained. To get there the hard way, we take the weighted sum of the squared differences between each group means and the overall mean, as follows:

In [17]: data

	company	score	overall_mean_x	group_mean	overall_mean_y
0	A	12.6	11.980952	12.542857	11.980952
1	A	12.0	11.980952	12.542857	11.980952
2	A	11.8	11.980952	12.542857	11.980952
3	A	11.9	11.980952	12.542857	11.980952
4	A	13.0	11.980952	12.542857	11.980952
5	A	12.5	11.980952	12.542857	11.980952
6	A	14.0	11.980952	12.542857	11.980952
7	B	10.0	11.980952	11.533333	11.980952
8	B	10.2	11.980952	11.533333	11.980952
9	B	10.0	11.980952	11.533333	11.980952
10	B	12.0	11.980952	11.533333	11.980952
11	B	14.0	11.980952	11.533333	11.980952
12	B	13.0	11.980952	11.533333	11.980952
13	C	10.1	11.980952	11.825000	11.980952
14	C	13.0	11.980952	11.825000	11.980952
15	C	13.4	11.980952	11.825000	11.980952
16	C	12.9	11.980952	11.825000	11.980952
17	C	8.9	11.980952	11.825000	11.980952
18	C	10.7	11.980952	11.825000	11.980952
19	C	13.6	11.980952	11.825000	11.980952
20	C	12.0	11.980952	11.825000	11.980952

```
In [18]: # compute Sum of Squares Model
ss_explained = sum((data['overall_mean_x'] - data['group_mean'])**2)
ss_explained

3.6069047619047776
```

The value can be found in the statsmodels table under `sum_sq` at the line `company`.

Degrees of freedom

I don't go into the degrees of freedom in this article, but we need them in further computation:

df1 = df of the explained part = number of groups — 1
df2 = df of the residual = number of observations — number of groups
In our example, df1 = 2 and df2 = 18.

Mean Squares

The statistical test that is central in ANOVA is the F-test. The null hypothesis states that the mean of all groups is equal, which implies that our model has no explanatory value and that we don't have proof for choosing one pizza company over another.

The alternative hypothesis states that at least one of the means is different, which would be a reason to go more in-depth and find out which company or companies are faster. We compute the Mean Squares as follows:

```
In [19]: # compute Mean Square Residual
n_groups = len(set(data['company']))
n_obs = data.shape[0]
df_residual = n_obs - n_groups
ms_residual = ss_residual / df_residual
ms_residual
```

2.1958597883597886

```
In [20]: # compute Mean Square Explained
df_explained = n_groups - 1
ms_explained = ss_explained / df_explained
ms_explained
```

1.8034523809523888

One-Way Annova Example with Bike Sharing Dataset

When there are more than 2 samples we use ANOVA testing. Anova is used to compare means across different groups.

For example ----- we want to check whether the average weighth of babies born in 3 different states are similar or different.

Before moving forward with any kind of hypothesis testing we should always have a question in our mind. And based on the question we decide the kind of hypothesis testing. And test our hypothesis against it.

Here we will work with the bike sharing data again. The question that we are asking here is ----- are the number of bike rentals similar or different in all 4 seasons.

###Pre-processing We have already looked at preprocessing so here let's dive in directly to hypothesis testing.

```
In [0]: #import the Libraries
import statsmodels.api as sm
from statsmodels.formula.api import ols
import random
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import scale
import warnings
from scipy import stats
%matplotlib inline

/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is deprecated. Use the functions in the public API at pandas.testing instead.
import pandas.util.testing as tm
```

```
In [0]: #read the data
df = pd.read_csv('/content/drive/My Drive/data_set/bike_sharing.csv')
```

```
In [0]: #drop datetime
df.drop(['datetime', 'atemp'],axis = 1,inplace=True)
```

```
In [0]: df['weather'].value_counts()
```

```
1    7192
2    2834
3     859
4         1
Name: weather, dtype: int64
```

We have only 1 record in 4th category. We will drop the records of 4th weather situation.

```
In [0]: df.drop(df[df['weather']==4].index,axis=0,inplace=True) #remove the records where weather == 4
```

```
In [0]: df.groupby('weather')['count'].describe() #groupby weather situation and check the description
```

	count	mean	std	min	25%	50%	75%	max
weather								
1	7192.0	205.236791	187.959566	1.0	48.0	161.0	305.0	977.0
2	2834.0	178.955540	168.366413	1.0	41.0	134.0	264.0	890.0
3	859.0	118.846333	138.581297	1.0	23.0	71.0	161.0	891.0

Clearly we can see that the means of 3 groups are very different. But are these differences stastically significant. We will use one-way anova to test whether this difference in mean is stastically significant or not.

So the question is does weather situation has any impact on the number of bikes rented or not.

```
In [0]: #perfrom one way annova using stats module from scipy library
#H0 : There is no difference in the mean
#H1 : There is a difference in the mean
#Alpha : 0.05

alpha = 0.05
Stats,p_value = stats.f_oneway(df['count'][df['weather']==1],
                                df['count'][df['weather']==2],
                                df['count'][df['weather']==3])

if p_value > alpha :
    print(f' Failed to reject null hypothesis \n Weather situation have no impact on bike rentals \n p-value
else:
    print(f' Reject null hypothesis \n Weather situation has impact on bike rentals \n p-value : {p_value}')
```

```
Reject null hypothesis
Weather situation has impact on bike rentals
p-value : 4.976448509904196e-43
```

Here our p-value is less than alpha. Which means that the weather situation impact the number of bike rentals.

Using one way anova we only know that the Means of the groups are not same. But we don't know which group mean are not same.

We use post-hoc test to find out which group mean are not equal.

###Tukey HSD Post-Hoc-Test

```
In [0]: #Use TukeyHSD to know which group mean are not similar.
from statsmodels.stats.multicomp import MultiComparison
mul_comp = MultiComparison(df['count'],df['weather'])
mul_result = mul_comp.tukeyhsd()
print(mul_result)
```

```
Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====
```

group1	group2	meandiff	p-adj	lower	upper	reject

1	2	-26.2813	0.001	-35.6159	-16.9466	True
1	3	-86.3905	0.001	-101.5842	-71.1968	True
2	3	-60.1092	0.001	-76.502	-43.7164	True

If you look at the last column. All the values are Reject = True. which means reject the null (Means are same) hypothesis . The mean of all the groups are significantly different.

##Two Way Anova Two way Anova is used to examine the influence of 2 different independent categorical variable on 1 dependent continuous variable.

In two way Annova we have 3 null hypothesis.

- 1. There is no effect on independent variable 1 on dependent variable.
- 2. There is no effect of independent variable 2 on dependent variable.
- 3. There is no interaction between variable 1 and variable 2.

Before ypothessis testing we perform a regression analysis using the two variables. We will go one step at a time and keep it simple to understand.

Let's examine whether Season and weather situation has any effect on bike rentals or not. We have 4 seasons and 3 weather situations.


```
In [0]: #check the description of groups of different weather situations
df.groupby('weather')['count'].describe()
```

	count	mean	std	min	25%	50%	75%	max
weather								
1	7192.0	205.236791	187.959566	1.0	48.0	161.0	305.0	977.0
2	2834.0	178.955540	168.366413	1.0	41.0	134.0	264.0	890.0
3	859.0	118.846333	138.581297	1.0	23.0	71.0	161.0	891.0

We had checked it before as well. And have proven that the means are stastically different in weather situations.

```
In [0]: #check the description of groups of different seasons
df.groupby('season')['count'].describe()
```

	count	mean	std	min	25%	50%	75%	max
season								
1	2685.0	116.325512	125.293931	1.0	24.0	78.0	164.0	801.0
2	2733.0	215.251372	192.007843	1.0	49.0	172.0	321.0	873.0
3	2733.0	234.417124	197.151001	1.0	68.0	195.0	347.0	977.0
4	2734.0	198.988296	177.622409	1.0	51.0	161.0	294.0	948.0

We can see that the means of the groups are different.

To Perform ANOVA Anlaysis we will first perform regression analysis.

```
In [0]: #Perfrom regression analysis with weather situation
model = ols('count ~ C(weather) * C(season)',df).fit() #fit the regression model
print(model.summary()) #print summary
```

OLS Regression Results						
=====						
Dep. Variable:	count	R-squared:	0.080			
Model:	OLS	Adj. R-squared:	0.079			
Method:	Least Squares	F-statistic:	85.66			
Date:	Sat, 18 Apr 2020	Prob (F-statistic):	8.63e-187			
Time:	07:31:15	Log-Likelihood:	-71587.			
No. Observations:	10885	AIC:	1.432e+05			
Df Residuals:	10873	BIC:	1.433e+05			
Df Model:	11					
Covariance Type:	nonrobust					
=====						
	coef	std err	t	P> t	[0.025	0.975]

Intercept	126.7817	4.146	30.582	0.000	118.656	134.908
C(weather)[T.2]	-19.9202	7.711	-2.583	0.010	-35.036	-4.804
C(weather)[T.3]	-65.5542	12.667	-5.175	0.000	-90.384	-40.724
C(season)[T.2]	109.9479	5.828	18.864	0.000	98.523	121.373
C(season)[T.3]	116.8017	5.731	20.379	0.000	105.567	128.036
C(season)[T.4]	82.7295	5.912	13.994	0.000	71.142	94.317
C(weather)[T.2]:C(season)[T.2]	-27.2939	10.906	-2.503	0.012	-48.672	-5.916
C(weather)[T.3]:C(season)[T.2]	-47.2691	17.669	-2.675	0.007	-81.904	-12.635
C(weather)[T.2]:C(season)[T.3]	7.1083	11.188	0.635	0.525	-14.823	29.039
C(weather)[T.3]:C(season)[T.3]	-21.4463	18.112	-1.184	0.236	-56.948	14.056
C(weather)[T.2]:C(season)[T.4]	5.1934	10.709	0.485	0.628	-15.799	26.185
C(weather)[T.3]:C(season)[T.4]	-9.4903	17.680	-0.537	0.591	-44.146	25.165
=====						
Omnibus:	1857.898	Durbin-Watson:	0.353			
Prob(Omnibus):	0.000	Jarque-Bera (JB):	3128.118			
Skew:	1.132	Prob(JB):	0.00			
Kurtosis:	4.331	Cond. No.	19.3			
=====						
Warnings:						
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.						

After performing regression analysis. We get 2 tables.

Table - 1 tells us whether the regression was significant or not. And table 2 tells us whether the variable is significant or not.

By looking at the table-1 we can see that the p-value related with f-statistics is very low. Which means the

regression was significant. Similarly when we look at the p-value associated with the t-statistic in table-2, we observe that the p-value is almost close to zero for most of the variables.

```
In [0]: #H0: There's no difference in mean of weather
#       There is No difference in Mean of Season
#       There is no difference in mean of Weather and Season combined

sm.stats.anova_lm(model) ##perform two way anova
```

	df	sum_sq	mean_sq	F	PR(>F)
C(weather)	2.0	6.337309e+06	3.168655e+06	104.818810	8.148093e-46
C(season)	3.0	2.158708e+07	7.195692e+06	238.032851	1.350921e-149
C(weather):C(season)	6.0	5.588352e+05	9.313920e+04	3.081036	5.150817e-03
Residual	10873.0	3.286889e+08	3.022983e+04	NaN	NaN

By looking at the p-values i.e the last columns we can see that most of the values are close to zero. So we can say that the means are significantly different.

Let's look into small Data Set & applying EDA as well as Chi-Square Test

```
In [1]: #Importing the required libraries
import numpy as np
import pandas as pd

import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')
```

executed in 1.22s, finished 17:04:01 2020-04-22

```
In [2]: #Reading the dataset
df_raw=pd.read_csv('QoSvsQoESyn.csv')
```

executed in 6ms, finished 17:04:01 2020-04-22

```
In [3]: # Creating a copy of dataset
copy_df=df_raw.copy(deep=True)
Copy_df = pd.DataFrame(df_raw, columns=['cellAccessibilityRankDesc'])
Copy_df.head()
```

executed in 123ms, finished 17:04:01 2020-04-22

	cellAccessibilityRankDesc
0	ACCEPTABLE
1	ACCEPTABLE
2	ACCEPTABLE
3	ACCEPTABLE
4	ACCEPTABLE

```
In [4]: # Display first five rows
copy_df.head()
```

executed in 105ms, finished 17:04:01 2020-04-22

	cellAccessibilityRankDesc	crmlnboundInteractionCount
0	ACCEPTABLE	1
1	ACCEPTABLE	1
2	ACCEPTABLE	1
3	ACCEPTABLE	1
4	ACCEPTABLE	1

What you can analyze from above Data Set & can you build an scalable ML Model from above information given & how you will interpret above Data Set to make it useful by undertsnading the columns names & making your own assumptions?

Let's work out some steps to discuss further

```
In [5]: # shape of the dataset
copy_df.shape
```

executed in 4ms, finished 17:04:02 2020-04-22

```
(3047, 2)
```

```
In [6]: #Checking the null values in the dataframe
copy_df.isnull().sum()
```

executed in 6ms, finished 17:04:03 2020-04-22

```
cellAccessibilityRankDesc      0
crmInboundInteractionCount      0
dtype: int64
```

```
In [7]: # information about dataset
copy_df.info()
```

executed in 9ms, finished 17:04:03 2020-04-22

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3047 entries, 0 to 3046
Data columns (total 2 columns):
cellAccessibilityRankDesc      3047 non-null object
crmInboundInteractionCount      3047 non-null int64
dtypes: int64(1), object(1)
memory usage: 47.7+ KB
```

```
In [8]: # Statistical description of dataframecopy_df.describe()
copy_df.describe()
```

executed in 13ms, finished 17:04:03 2020-04-22

	crmInboundInteractionCount
count	3047.000000
mean	0.985560
std	0.780879
min	0.000000
25%	0.000000
50%	1.000000
75%	2.000000
max	2.000000

```
In [9]: # Statistical description of dataframe
copy_df.describe().T #transforming the above code to have a better view of statistical summary
```

executed in 18ms, finished 17:04:03 2020-04-22

	count	mean	std	min	25%	50%	75%	max
crmInboundInteractionCount	3047.0	0.98556	0.780879	0.0	0.0	1.0	2.0	2.0

```
In [10]: # Extraxting a unique values of 'cellAccessibilityRankDesc' Column
a=copy_df['cellAccessibilityRankDesc'].unique()
print(a)
len(a)
```

executed in 6ms, finished 17:04:04 2020-04-22

```
['ACCEPTABLE' 'EXCELLENT' 'GOOD']
```

```
3
```

```
In [11]: #Finding the count of QoS
copy_df['cellAccessibilityRankDesc'].value_counts()
```

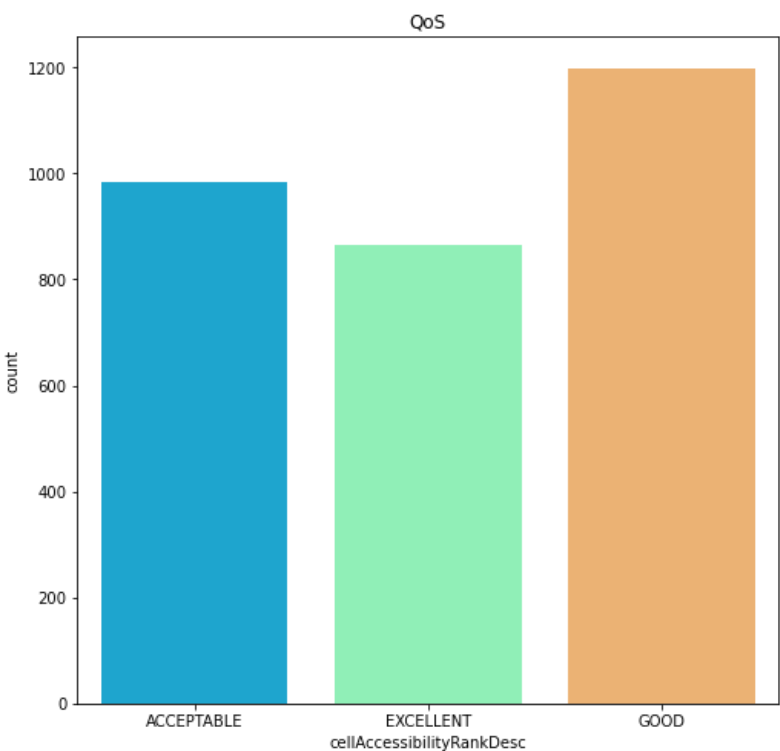
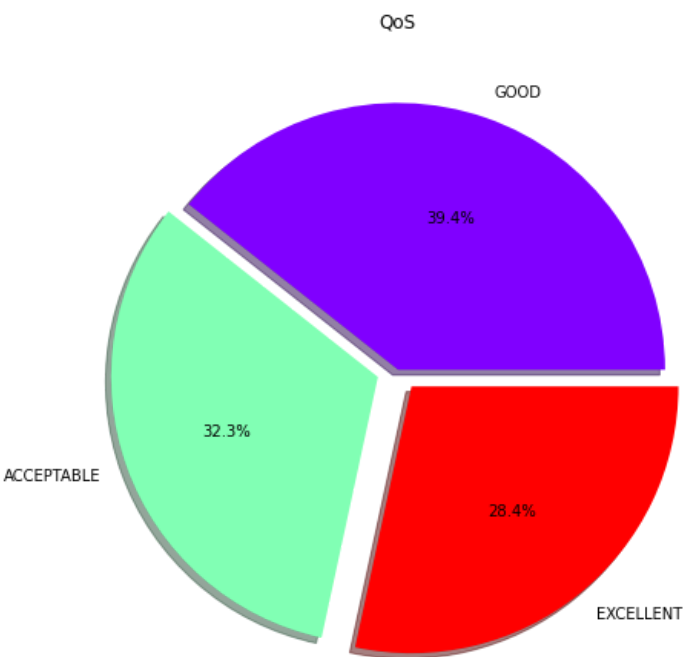
executed in 13ms, finished 17:04:04 2020-04-22

GOOD	1199
ACCEPTABLE	983
EXCELLENT	865
Name: cellAccessibilityRankDesc, dtype: int64	

As we can observe above that there are three unique values in 'cellAccessibilityRankDesc'.

```
In [12]: #plotting the count of QoS
f,ax=plt.subplots(1,2,figsize=(18,8))
copy_df['cellAccessibilityRankDesc'].value_counts().plot.pie(explode=[0,0.08,0.08],autopct='%1.1f%%',ax=ax[0])
ax[0].set_title('QoS')
ax[0].set_ylabel('')
sns.countplot('cellAccessibilityRankDesc',data=copy_df,ax=ax[1], palette='rainbow')
ax[1].set_title('QoS')
plt.show()
```

executed in 265ms, finished 17:04:04 2020-04-22



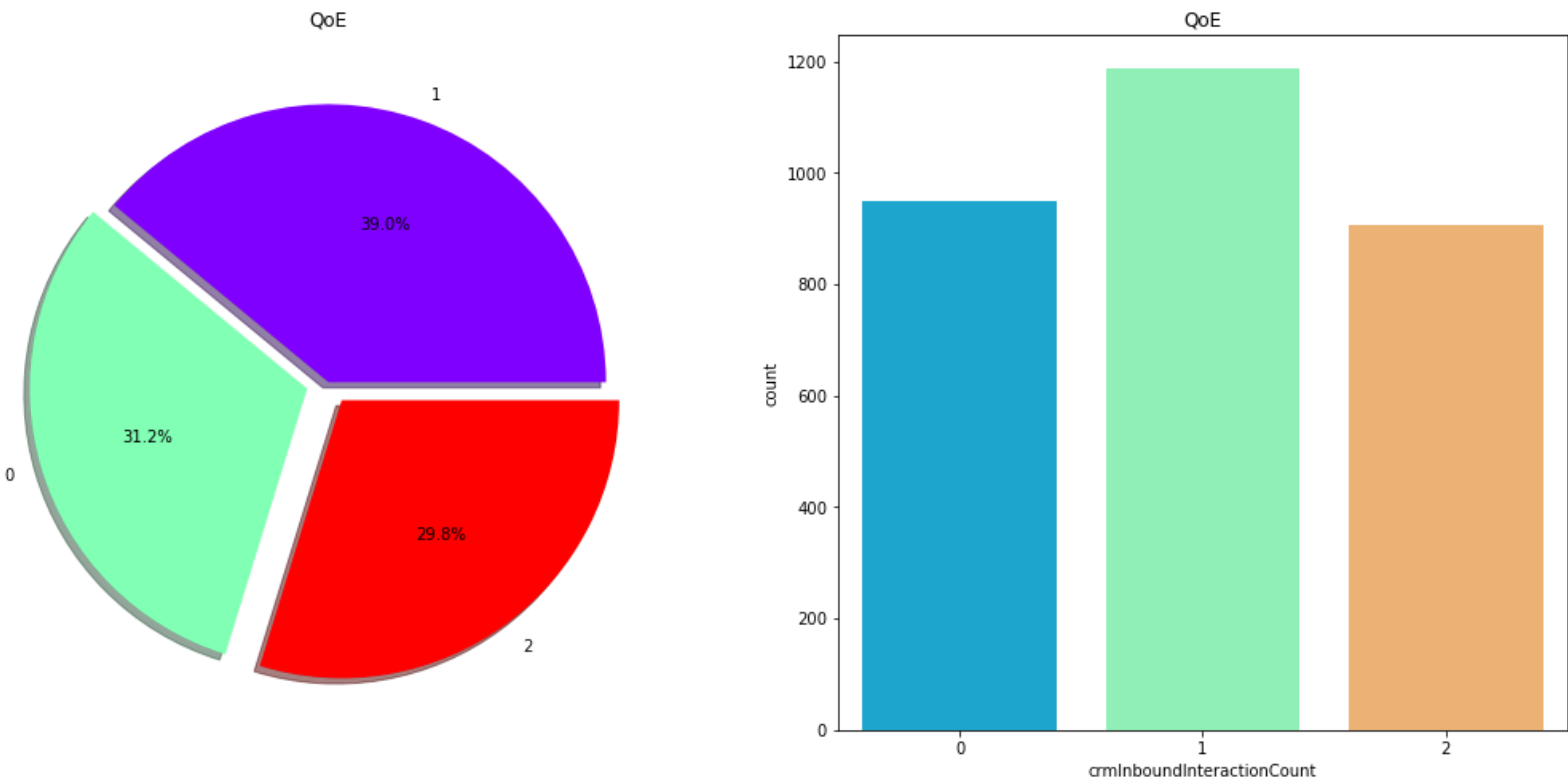
```
In [13]: #Finding the count of the user experiences
copy_df['crmInboundInteractionCount'].value_counts()
```

executed in 7ms, finished 17:04:04 2020-04-22

1	1189
0	951
2	907
Name: crmInboundInteractionCount, dtype: int64	

```
In [14]: #Plotting the count of QoE
f,ax=plt.subplots(1,2,figsize=(18,8))
copy_df['crmInboundInteractionCount'].value_counts().plot.pie(explode=[0,0.08,0.08],autopct='%1.1f%%',ax=ax
ax[0].set_title('QoE')
ax[0].set_ylabel('')
sns.countplot('crmInboundInteractionCount',data=copy_df,ax=ax[1], palette='rainbow')
ax[1].set_title('QoE')
plt.show()
```

executed in 203ms, finished 17:04:05 2020-04-22



```
In [15]: #Finding the unique values of user experience for each parameter of QoS
copy_df.groupby('cellAccessibilityRankDesc')['crmInboundInteractionCount'].unique()
```

executed in 10ms, finished 17:04:05 2020-04-22

```
cellAccessibilityRankDesc
ACCEPTABLE    [1, 2, 0]
EXCELLENT     [0, 1, 2]
GOOD          [2, 1, 0]
Name: crmInboundInteractionCount, dtype: object
```

```
In [16]: #Finding the count of user experience in each service quality
copy_df.groupby(['cellAccessibilityRankDesc', 'crmInboundInteractionCount']).size()
```

executed in 8ms, finished 17:04:05 2020-04-22

cellAccessibilityRankDesc	crmInboundInteractionCount	
ACCEPTABLE	0	55
	1	884
	2	44
EXCELLENT	0	767
	1	33
	2	65
GOOD	0	129
	1	272
	2	798
dtype: int64		

```
In [17]: #Drawing crosstab for the QoS and QoE for easily understanding as per convenience
pd.crosstab(copy_df['cellAccessibilityRankDesc'],df_raw['crmInboundInteractionCount'],margins=True).style.b
```

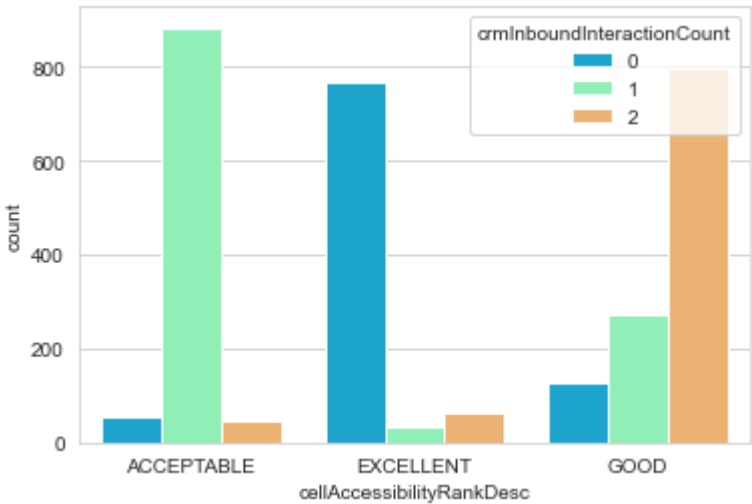
executed in 218ms, finished 17:04:06 2020-04-22

crmInboundInteractionCount	0	1	2	All
cellAccessibilityRankDesc				
ACCEPTABLE	55	884	44	983
EXCELLENT	767	33	65	865
GOOD	129	272	798	1199
All	951	1189	907	3047

```
In [18]: sns.set_style('whitegrid')
sns.countplot(x='cellAccessibilityRankDesc', hue='crmInboundInteractionCount', data=copy_df, palette='rainb
```

executed in 285ms, finished 17:04:06 2020-04-22

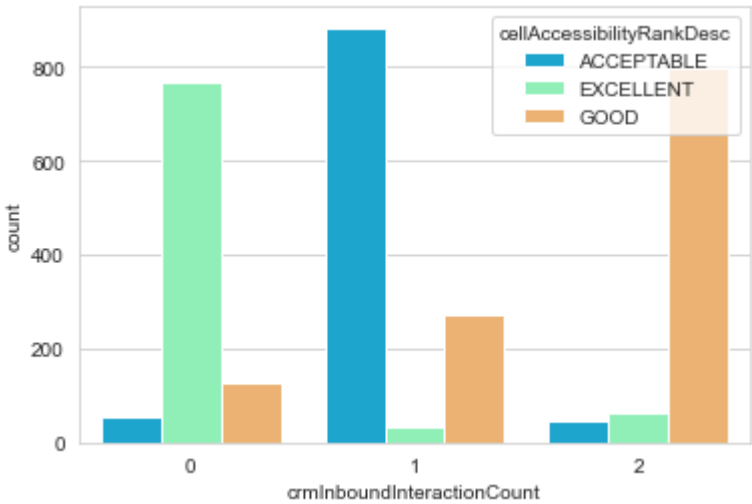
<matplotlib.axes._subplots.AxesSubplot at 0x6ffffb12888>



```
In [19]: sns.set_style('whitegrid')
sns.countplot(x='crmInboundInteractionCount', hue='cellAccessibilityRankDesc', data=copy_df, palette='rainb
```

executed in 220ms, finished 17:04:15 2020-04-22

<matplotlib.axes._subplots.AxesSubplot at 0x6f87755648>



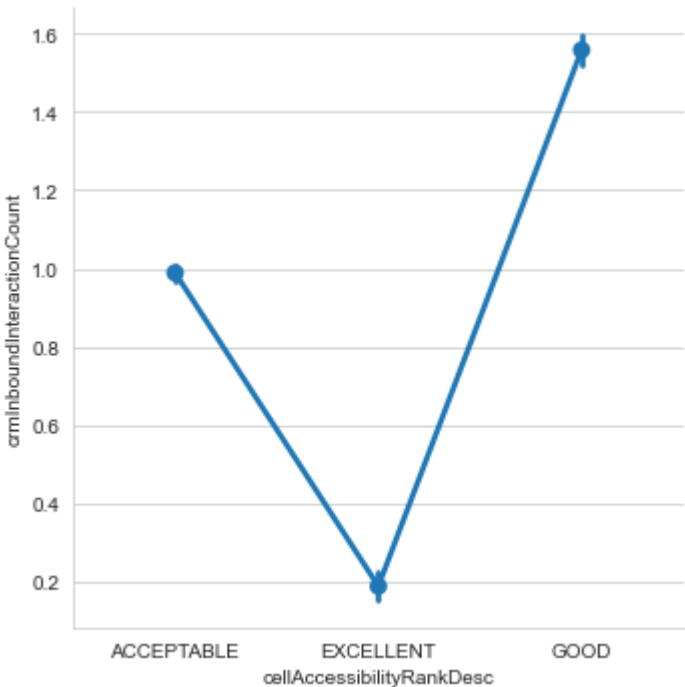

```
In [20]: #Finding the average value of user experience for each of the service qualities
copy_df[['cellAccessibilityRankDesc', 'crmInboundInteractionCount']].groupby(['cellAccessibilityRankDesc'])
```

executed in 11ms, finished 17:04:15 2020-04-22

	crmInboundInteractionCount
cellAccessibilityRankDesc	
GOOD	1.557965
ACCEPTABLE	0.988810
EXCELLENT	0.188439

```
In [21]: #Using factorplot to find the factor for the user rating for each of the service quality
sns.factorplot('cellAccessibilityRankDesc', 'crmInboundInteractionCount', data=copy_df)
plt.show()
```

executed in 572ms, finished 17:04:16 2020-04-22



Let's apply one-hot encoding to 'cellAccessibilityRankDesc',so that our model can understand it easily without being confused

```
In [22]: copy_df=pd.get_dummies(df_raw,drop_first=True)
```

executed in 9ms, finished 17:04:16 2020-04-22

```
In [23]: copy_df
```

executed in 10ms, finished 17:04:16 2020-04-22

	crmInboundInteractionCount	cellAccessibilityRankDesc_EXCELLENT	cellAccessibilityRankDesc_GOOD
0	1	0	0
1	1	0	0
2	1	0	0
3	1	0	0
4	1	0	0
...
3042	0	0	1
3043	0	0	1
3044	0	0	1
3045	0	0	1
3046	0	0	1

3047 rows × 3 columns

As you can observe from above output we losted one of the unique value in 'cellAccessibilityRankDesc' which is 'Acceptable',so one hot encoding doesn't work out.

Figure out on your own why this happen & what is it called?

If you didn't get the answer do reach me out on vivekchaudhary0612@gmail.com (<mailto:vivekchaudhary0612@gmail.com>),will be happy to share.

So,how can we make it happen in a proper format without lossing any of the unique value,let's have a look on it.

```
In [24]: # Creating a copy of dataset
copy_df1=df_raw.copy(deep=True)
Copy_df1 = pd.DataFrame(df_raw, columns=['cellAccessibilityRankDesc'])
Copy_df1.head()
```

executed in 9ms, finished 17:04:36 2020-04-22

	cellAccessibilityRankDesc
0	ACCEPTABLE
1	ACCEPTABLE
2	ACCEPTABLE
3	ACCEPTABLE
4	ACCEPTABLE

```
In [25]: # applying get dummy method
dum_df = pd.get_dummies(Copy_df, columns=["cellAccessibilityRankDesc"] )
```

executed in 6ms, finished 17:04:37 2020-04-22

```
In [26]: dum_df
```

executed in 9ms, finished 17:04:37 2020-04-22

	cellAccessibilityRankDesc_ACCEPTABLE	cellAccessibilityRankDesc_EXCELLENT	cellAccessibilityRankDesc_GOOD
0	1	0	0
1	1	0	0
2	1	0	0
3	1	0	0
4	1	0	0
...
3042	0	0	1
3043	0	0	1
3044	0	0	1
3045	0	0	1
3046	0	0	1

3047 rows × 3 columns

```
In [27]: copy_df1.drop('cellAccessibilityRankDesc',axis=1,inplace=True)
copy_df1.head()
```

executed in 7ms, finished 17:04:38 2020-04-22

crmlInboundInteractionCount	
0	1
1	1
2	1
3	1
4	1

```
In [28]: new_data=dum_df.join(df_raw)
new_data.head()
```

executed in 24ms, finished 17:04:38 2020-04-22

	cellAccessibilityRankDesc_ACCEPTABLE	cellAccessibilityRankDesc_EXCELLENT	cellAccessibilityRankDesc_GOOD	cellAccess
0	1	0	0	ACCEPTAE
1	1	0	0	ACCEPTAE
2	1	0	0	ACCEPTAE
3	1	0	0	ACCEPTAE
4	1	0	0	ACCEPTAE

As you can see above we have used simple technique to make that happen & instead you can use others encoding tecjniques,just to show you that we can't apply one-hot encoding everytime & everywhere.

So,try to understand the problem statement based on column names & see hwat assumption else which business you can relate it & work out with your ML Model!!

Take it as a challenge can we build a small model to predict something out of this Data Set ,YES OR NO!!

Mail us your answer & get in touch for more discussion at vivekchaudhary0612@gmail.com (<mailto:vivekchaudhary0612@gmail.com>).

Using Chi-Square Test

Before moving further let's create assumption:

Ho(Null hypotheisis)=Two variables are independent

Ha(Alternative hypothesis)=Two variables are not independent

```
In [29]: df_raw.head()
```

executed in 7ms, finished 17:04:47 2020-04-22

	cellAccessibilityRankDesc	crmInboundInteractionCount
0	ACCEPTABLE	1
1	ACCEPTABLE	1
2	ACCEPTABLE	1
3	ACCEPTABLE	1
4	ACCEPTABLE	1

```
In [30]: rank_desk=pd.crosstab(index=df_raw['cellAccessibilityRankDesc'],columns=df_raw['crmInboundInteractionCount'])
```

executed in 18ms, finished 17:04:47 2020-04-22

```
In [31]: rank_desk #before applying particular test we have to Look for Contingency table
```

executed in 7ms, finished 17:04:48 2020-04-22

crmInboundInteractionCount	0	1	2
cellAccessibilityRankDesc			
ACCEPTABLE	55	884	44
EXCELLENT	767	33	65
GOOD	129	272	798

Degrees of freedom for contingency table is given as $(r-1) * (c-1)$ where r,c are rows and columns. Here $df = (3-1)(3-1)=2*2=4$,so here degree of freedom is 4.

So,what significant Degree of freedom plays,common search it out ,please don't be lazy!!

```
In [32]: rank_desk.iloc[0].values #we can observe the particular value at every index
```

executed in 5ms, finished 17:04:48 2020-04-22

```
array([ 55, 884,  44], dtype=int64)
```

```
In [33]: from scipy import stats #import stats package
(chi2, p, dof,_) = stats.chi2_contingency([rank_desk.iloc[0].values,rank_desk.iloc[1].values,rank_desk.iloc[2].values])
```

executed in 19ms, finished 17:04:49 2020-04-22

```
In [34]: print ("chi2      : " ,chi2)
print ("p-value   : " ,p)
print ("Degree of Freedom : " ,dof)
```

executed in 3ms, finished 17:04:49 2020-04-22

```
chi2      : 3192.2255875828437
p-value   : 0.0
Degree of Freedom : 4
```

A p-value of less than 0.05 implies significance and that of less than 0.01 implies high significance. Therefore $p=0.0000$ implies high significance. Hence we can conclude that this two variables are highle significant

Now question that may be come around is:

what is degree of freedom?

why we always compare p values with 0.05,why not 0.06 or 0.07,common give a thought & research about the same.

We can also say that as the degrees of freedom increase Chi-Square distribution approximates to normal distribution.

So,take this challenge & find the answer,do mail us at vivekchaudhary0612@gmail.com (<mailto:vivekchaudhary0612@gmail.com>).

When to use chi-square?

When using a two categorical feature and trying to conclude to accept or reject null hypothesis chi-Square test works best. The Chi-Square test is most useful when analyzing cross tabulations of survey response data. Because cross tabulations reveal the frequency and percentage of responses to questions by various segments or categories of respondents (gender, profession, education level, etc.)

In Simple words,Let's consider a scenario where we need to determine the relationship between the independent category feature (predictor) and dependent category feature(response). In feature selection, we aim to select the features which are highly dependent on the response.(source::Google)

Limitations of chi-Square

Can not be used when samples are matched or related.

It wont give much information about strength of the relationship.

Now we have gone throught some interesting stuff & wish you have learned a lot,let's have look for example

```
In [35]: import pandas as pd
import numpy as np
```

executed in 3ms, finished 17:04:52 2020-04-22

```
In [38]: data=pd.read_csv('zomato.csv')
```

executed in 4.75s, finished 17:05:06 2020-04-22

```
In [39]: print("Percentage null or na values in df")
((data.isnull() | data.isna()).sum() * 100 / data.index.size).round(2)
```

executed in 95ms, finished 17:05:07 2020-04-22

```
Percentage null or na values in df

url          0.00
address      0.00
name         0.00
online_order 0.00
book_table   0.00
rate         15.03
votes        0.00
phone        2.34
location     0.04
rest_type    0.44
dish_liked   54.29
cuisines     0.09
approx_cost(for two people) 0.67
reviews_list 0.00
menu_item    0.00
listed_in(type) 0.00
listed_in(city) 0.00
dtype: float64
```

As we can have a look on Data Set of zomato food price prediction & let's work on some basic data cleaning steps & let's apply some encoding techniques with some statistical significance.

```
In [40]: data.info() #checking different type of data types particulr feature belong to.
```

executed in 51ms, finished 17:05:08 2020-04-22

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51717 entries, 0 to 51716
Data columns (total 17 columns):
url          51717 non-null object
address      51717 non-null object
name         51717 non-null object
online_order 51717 non-null object
book_table   51717 non-null object
rate         43942 non-null object
votes        51717 non-null int64
phone        50509 non-null object
location     51696 non-null object
rest_type    51490 non-null object
dish_liked   23639 non-null object
cuisines     51672 non-null object
approx_cost(for two people) 51371 non-null object
reviews_list 51717 non-null object
menu_item    51717 non-null object
listed_in(type) 51717 non-null object
listed_in(city) 51717 non-null object
dtypes: int64(1), object(16)
memory usage: 6.7+ MB
```

```
In [41]: #Deleting Unnnecessary Columns
data=data.drop(['url','dish_liked','phone'],axis=1) #Dropping the column "dish_liked", "phone", "url" and s
```

executed in 21ms, finished 17:05:09 2020-04-22

In [42]: data.head()

executed in 16ms, finished 17:05:09 2020-04-22

	address	name	online_order	book_table	rate	votes	location	rest_type	cuisines	approx_cost(for two people)	reviews_list
0	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	Yes	Yes	4.1/5	775	Banashankari	Casual Dining	North Indian, Mughlai, Chinese	800	['Rai 'RAT beau to ...
1	2nd Floor, 80 Feet Road, Near Big Bazaar, 6th ...	Spice Elephant	Yes	No	4.1/5	787	Banashankari	Casual Dining	Chinese, North Indian, Thai	800	['Rai 'RAT been din...
2	1112, Next to KIMS Medical College, 17th Cross...	San Churro Cafe	Yes	No	3.8/5	918	Banashankari	Cafe, Casual Dining	Cafe, Mexican, Italian	800	['Rai "RAT Ambi not th
3	1st Floor, Annakuteera, 3rd Stage, Banashankar...	Addhuri Udupi Bhojana	No	No	3.7/5	88	Banashankari	Quick Bites	South Indian, North Indian	300	['Rai "RAT Grea prope
4	10, 3rd Floor, Lakshmi Associates, Gandhi Baza...	Grand Village	No	No	3.8/5	166	Basavanagudi	Casual Dining	North Indian, Rajasthani	600	['Rai 'RAT good resta

In [43]: *#Removing the Duplicates*
data.duplicated().sum()
data.drop_duplicates(inplace=True)

executed in 3.31s, finished 17:05:13 2020-04-22

In [44]: *#Remove the NaN values from the dataset*
data.isnull().sum()
data.dropna(how='any',inplace=True)
data.info() *#.info() function is used to get a concise summary of the dataframe*

executed in 115ms, finished 17:05:13 2020-04-22

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 43499 entries, 0 to 51716
Data columns (total 14 columns):
address                43499 non-null object
name                   43499 non-null object
online_order           43499 non-null object
book_table             43499 non-null object
rate                   43499 non-null object
votes                  43499 non-null int64
location               43499 non-null object
rest_type              43499 non-null object
cuisines               43499 non-null object
approx_cost(for two people) 43499 non-null object
reviews_list           43499 non-null object
menu_item              43499 non-null object
listed_in(type)        43499 non-null object
listed_in(city)        43499 non-null object
dtypes: int64(1), object(13)
memory usage: 5.0+ MB
```

In [45]: *#Reading Column Names*
data.columns

executed in 33ms, finished 17:05:13 2020-04-22

```
Index(['address', 'name', 'online_order', 'book_table', 'rate', 'votes',
      'location', 'rest_type', 'cuisines', 'approx_cost(for two people)',
      'reviews_list', 'menu_item', 'listed_in(type)', 'listed_in(city)'],
      dtype='object')
```



```
In [46]: #Changing the column names
data= data.rename(columns={'approx_cost(for two people)': 'cost', 'listed_in(type)': 'type',
                           'listed_in(city)': 'city'})

data.columns
```

executed in 20ms, finished 17:05:14 2020-04-22

```
Index(['address', 'name', 'online_order', 'book_table', 'rate', 'votes',
      'location', 'rest_type', 'cuisines', 'cost', 'reviews_list',
      'menu_item', 'type', 'city'],
      dtype='object')
```

```
In [47]: data['cost'].unique() #Looking the unique values of cost column
```

executed in 8ms, finished 17:05:15 2020-04-22

```
array(['800', '300', '600', '700', '550', '500', '450', '650', '400',
      '900', '200', '750', '150', '850', '100', '1,200', '350', '250',
      '950', '1,000', '1,500', '1,300', '199', '80', '1,100', '160',
      '1,600', '230', '130', '1,700', '1,400', '1,350', '2,200', '2,000',
      '1,800', '1,900', '180', '330', '2,500', '2,100', '3,000', '2,800',
      '3,400', '50', '40', '1,250', '3,500', '4,000', '2,400', '2,600',
      '1,450', '70', '3,200', '560', '240', '360', '6,000', '1,050',
      '2,300', '4,100', '120', '5,000', '3,700', '1,650', '2,700',
      '4,500'], dtype=object)
```

```
In [48]: #Reading Rate of dataset
data['rate'].unique()
```

executed in 7ms, finished 17:05:15 2020-04-22

```
array(['4.1/5', '3.8/5', '3.7/5', '3.6/5', '4.6/5', '4.0/5', '4.2/5',
      '3.9/5', '3.1/5', '3.0/5', '3.2/5', '3.3/5', '2.8/5', '4.4/5',
      '4.3/5', 'NEW', '2.9/5', '3.5/5', '2.6/5', '3.8 /5', '3.4/5',
      '4.5/5', '2.5/5', '2.7/5', '4.7/5', '2.4/5', '2.2/5', '2.3/5',
      '3.4 /5', '-', '3.6 /5', '4.8/5', '3.9 /5', '4.2 /5', '4.0 /5',
      '4.1 /5', '3.7 /5', '3.1 /5', '2.9 /5', '3.3 /5', '2.8 /5',
      '3.5 /5', '2.7 /5', '2.5 /5', '3.2 /5', '2.6 /5', '4.5 /5',
      '4.3 /5', '4.4 /5', '4.9/5', '2.1/5', '2.0/5', '1.8/5', '4.6 /5',
      '4.9 /5', '3.0 /5', '4.8 /5', '2.3 /5', '4.7 /5', '2.4 /5',
      '2.1 /5', '2.2 /5', '2.0 /5', '1.8 /5'], dtype=object)
```

```
In [49]: data.rate = data.rate.replace("NEW", np.nan)
data.dropna(how = 'any', inplace = True)
```

executed in 46ms, finished 17:05:15 2020-04-22

```
In [50]: data.rate = data.rate.replace("-", np.nan)
data.dropna(how = 'any', inplace = True)
```

executed in 43ms, finished 17:05:16 2020-04-22

```
In [51]: X = data
X.rate = X.rate.astype(str)
X.rate = X.rate.apply(lambda x: x.replace('/5',''))
X.rate = X.rate.apply(lambda x: float(x))
X.head()
```

executed in 68ms, finished 17:05:16 2020-04-22

	address	name	online_order	book_table	rate	votes	location	rest_type	cuisines	cost	reviews_list
0	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	Yes	Yes	4.1	775	Banashankari	Casual Dining	North Indian, Mughlai, Chinese	800	['Rated 4.0', 'RATED\n A beautiful place to ...
1	2nd Floor, 80 Feet Road, Near Big Bazaar, 6th ...	Spice Elephant	Yes	No	4.1	787	Banashankari	Casual Dining	Chinese, North Indian, Thai	800	['Rated 4.0', 'RATED\n Had been here for din...
2	1112, Next to KIMS Medical College, 17th Cross...	San Churro Cafe	Yes	No	3.8	918	Banashankari	Cafe, Casual Dining	Cafe, Mexican, Italian	800	['Rated 3.0', 'RATED\n Ambience is not that ...
3	1st Floor, Annakuteera, 3rd Stage, Banashankar...	Addhuri Udupi Bhojana	No	No	3.7	88	Banashankari	Quick Bites	South Indian, North Indian	300	['Rated 4.0', 'RATED\n Great food and proper...
4	10, 3rd Floor, Lakshmi Associates, Gandhi Baza...	Grand Village	No	No	3.8	166	Basavanagudi	Casual Dining	North Indian, Rajasthani	600	['Rated 4.0', 'RATED\n Very good restaurant ...

```
In [52]: data['rate'].unique()
```

executed in 7ms, finished 17:05:16 2020-04-22

```
array([4.1, 3.8, 3.7, 3.6, 4.6, 4. , 4.2, 3.9, 3.1, 3. , 3.2, 3.3, 2.8,
       4.4, 4.3, 2.9, 3.5, 2.6, 3.4, 4.5, 2.5, 2.7, 4.7, 2.4, 2.2, 2.3,
       4.8, 4.9, 2.1, 2. , 1.8])
```

```
In [53]: data.online_order = data.online_order.apply(lambda X : 0 if X == 'No' else 1)
data.book_table=data.book_table.apply(lambda X : 0 if X == 'No' else 1)
```

executed in 46ms, finished 17:05:16 2020-04-22

In [54]: data.head()

executed in 18ms, finished 17:05:16 2020-04-22

	address	name	online_order	book_table	rate	votes	location	rest_type	cuisines	cost	reviews_list
0	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	1	1	4.1	775	Banashankari	Casual Dining	North Indian, Mughlai, Chinese	800	['(Rated 4.0', 'RATED\n A beautiful place to ...
1	2nd Floor, 80 Feet Road, Near Big Bazaar, 6th ...	Spice Elephant	1	0	4.1	787	Banashankari	Casual Dining	Chinese, North Indian, Thai	800	['(Rated 4.0', 'RATED\n Had been here for din...
2	1112, Next to KIMS Medical College, 17th Cross...	San Churro Cafe	1	0	3.8	918	Banashankari	Cafe, Casual Dining	Cafe, Mexican, Italian	800	['(Rated 3.0', 'RATED\n Ambience is not that ...
3	1st Floor, Annakuteera, 3rd Stage, Banashankar...	Addhuri Udupi Bhojana	0	0	3.7	88	Banashankari	Quick Bites	South Indian, North Indian	300	['(Rated 4.0', 'RATED\n Great food and proper...
4	10, 3rd Floor, Lakshmi Associates, Gandhi Baza...	Grand Village	0	0	3.8	166	Basavanagudi	Casual Dining	North Indian, Rajasthani	600	['(Rated 4.0', 'RATED\n Very good restaurant ...

```
#Encode the input Variables
def Encode(data):
    for column in data.columns[~data.columns.isin(['rate', 'cost', 'votes'])]:
        data[column] = data[column].factorize()[0]
    return data

data_en = Encode(data.copy())
```

executed in 1.43s, finished 17:05:18 2020-04-22

In [56]: data_en.head(2)

executed in 14ms, finished 17:05:18 2020-04-22

	address	name	online_order	book_table	rate	votes	location	rest_type	cuisines	cost	reviews_list	menu_item
0	0	0	0	0	4.1	775	0	0	0	800	0	0
1	1	1	0	1	4.1	787	0	0	1	800	1	0

```
#data_en.isnull().sum()
data=data.dropna(axis=1,how='all')
```

executed in 133ms, finished 17:05:18 2020-04-22

In [58]: data.info()

executed in 138ms, finished 17:05:18 2020-04-22

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 41237 entries, 0 to 51716
Data columns (total 14 columns):
address      41237 non-null object
name         41237 non-null object
online_order 41237 non-null int64
book_table   41237 non-null int64
rate         41237 non-null float64
votes        41237 non-null int64
location     41237 non-null object
rest_type    41237 non-null object
cuisines     41237 non-null object
cost         41237 non-null object
reviews_list 41237 non-null object
menu_item    41237 non-null object
type         41237 non-null object
city         41237 non-null object
dtypes: float64(1), int64(3), object(10)
memory usage: 4.7+ MB
```

In [59]: `from sklearn.feature_selection import chi2 #importing chi2`

executed in 2.31s, finished 17:05:21 2020-04-22

In [60]: `data_en.isnull().sum() #Looking into null values`

executed in 17ms, finished 17:05:21 2020-04-22

```
address      0
name         0
online_order 0
book_table   0
rate         0
votes        0
location     0
rest_type    0
cuisines     0
cost         0
reviews_list 0
menu_item    0
type         0
city         0
dtype: int64
```

In [61]: `data_en.isnull().sum() #deleting null values in rate column`
`data_en.dropna(how='any',inplace=True)`

executed in 133ms, finished 17:05:21 2020-04-22

In [62]: `data_en.isnull().sum() #checking again is there any null values present or not`

executed in 134ms, finished 17:05:21 2020-04-22

```
address      0
name         0
online_order 0
book_table   0
rate         0
votes        0
location     0
rest_type    0
cuisines     0
cost         0
reviews_list 0
menu_item    0
type         0
city         0
dtype: int64
```

In [63]: `X = data_en.drop('cost',axis=1) #dividing into independent variable & target variable which is cost`
`y = data_en['cost']`

executed in 213ms, finished 17:05:21 2020-04-22

In [64]: `chi_scores = chi2(X,y)`

executed in 514ms, finished 17:05:22 2020-04-22

In [65]: `chi_scores #In below output first array represent chi2 values & second array represent p values`

executed in 5ms, finished 17:05:22 2020-04-22

```
(array([5.62117186e+05, 1.00339961e+06, 1.91456379e+03, 3.00648412e+03,
        4.01775036e+02, 2.03865805e+07, 7.70550795e+03, 1.32596277e+05,
        2.78241858e+06, 6.24893941e+06, 2.30839189e+06, 1.11926352e+03,
        1.18575758e+03]),
 array([0.00000000e+000, 0.00000000e+000, 0.00000000e+000, 0.00000000e+000,
        3.09307669e-051, 0.00000000e+000, 0.00000000e+000, 0.00000000e+000,
        0.00000000e+000, 0.00000000e+000, 0.00000000e+000, 9.82444675e-194,
        2.01298388e-207]))
```

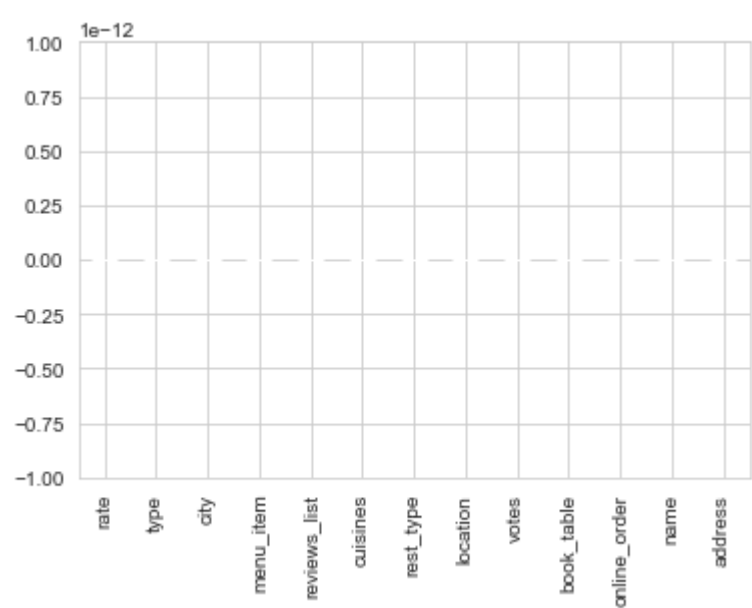
In [66]: `p_values = pd.Series(chi_scores[1],index = X.columns)`
`p_values.sort_values(ascending = False , inplace = True)`

executed in 102ms, finished 17:05:22 2020-04-22

```
In [67]: p_values.plot.bar()
```

executed in 430ms, finished 17:05:22 2020-04-22

<matplotlib.axes._subplots.AxesSubplot at 0x6f9bd05808>



Since 'rate' has higher the p-value, it says that this variables is independent of the repsons and can not be considered for model training

But point to be noted statistics will give likely to haapen result not 100% conformation & still at the end whether to choose 'rate' or not during model training totally depends on your strong problem statement understanding & based upon your commensence.

Remember key skill to solve any problem statement as a Data Scientist is Research,Commensence & your statistics knowledge.

Chi-Square Example (E-Commerece Dataset)

```
In [0]: import pandas as pd #import pandas
import numpy as np #import numpy
import matplotlib.pyplot as plt #import matplotlib
from scipy.stats import chi2_contingency #import chi2_contingency for chi2 independency tet
from matplotlib import style
style.use('ggplot')
```

```
In [0]: df = pd.read_csv('/content/drive/My Drive/data_set/E-commerce.csv') #read the data
```

```
In [0]: df.shape #check shape of the data
```

(23472, 9)

```
In [0]: df.head() #check head of the data
```

	Unnamed: 0	Clothing ID	Age	Rating	Recommended IND	Positive Feedback Count	Division Name	Department Name	Class Name
0	0	767	33	4	1	0	Initmates	Intimate	Intimates
1	1	1080	34	5	1	4	General	Dresses	Dresses
2	2	1077	60	3	0	0	General	Dresses	Dresses
3	3	1049	50	5	1	0	General Petite	Bottoms	Pants
4	4	847	47	5	1	6	General	Tops	Blouses

```
In [0]: df.drop('Unnamed: 0',axis=1,inplace=True) #drop Unnamed: 0 column
```

```
In [0]: df.columns #check columns
```

```
Index(['Clothing ID', 'Age', 'Rating', 'Recommended IND',
      'Positive Feedback Count', 'Division Name', 'Department Name',
      'Class Name'],
      dtype='object')
```

```
In [0]: df.isnull().sum()
```

```
Clothing ID      0
Age              0
Rating           0
Recommended IND  0
Positive Feedback Count  0
Division Name     0
Department Name  0
Class Name       0
dtype: int64
```

Here we will work only with 2 columns Recommended IND and Ratings.

We want to check whether the website recommendations are independent of ratings or not.

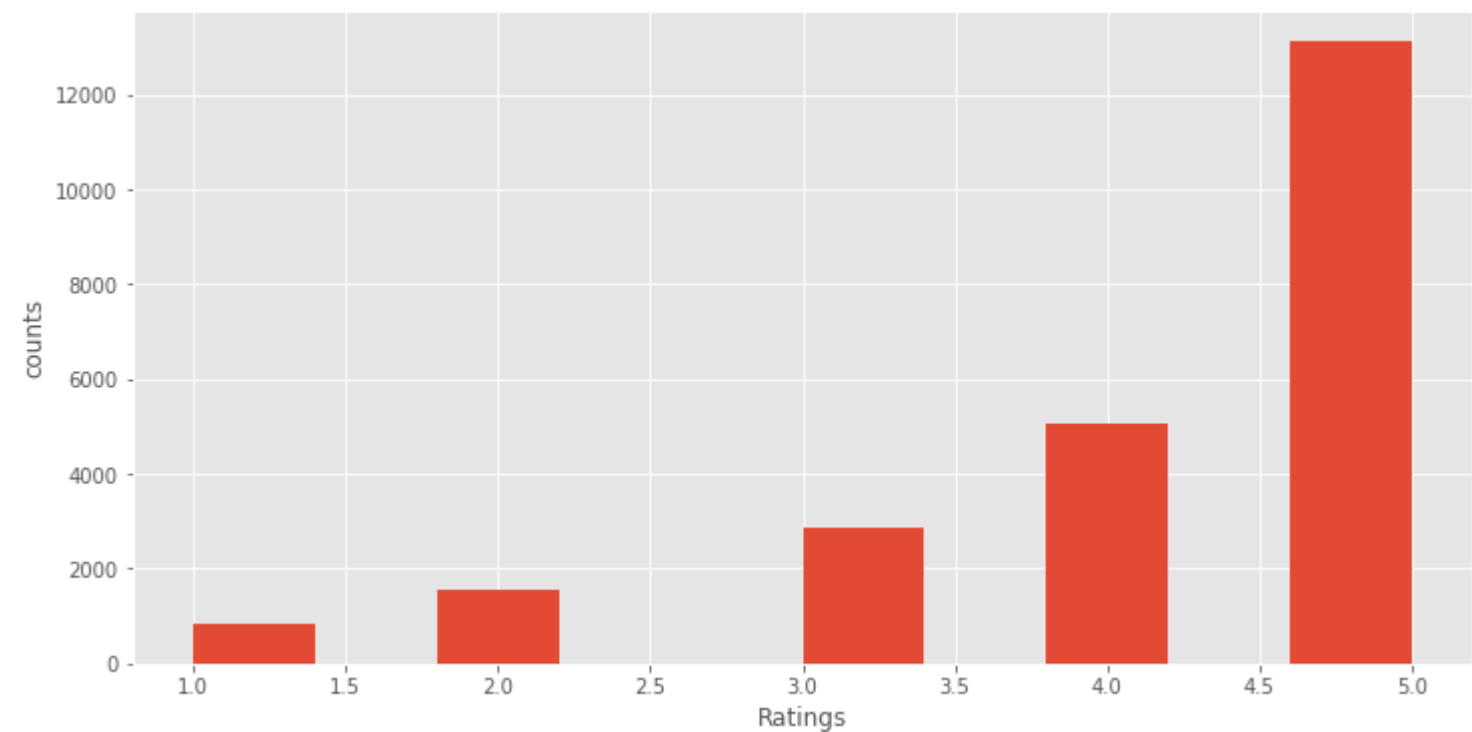
```
In [0]: df_1 = df[['Rating','Recommended IND']] #store rating and Recommended IND in df_1
```

```
In [0]: df_1.head() #check head
```

	Rating	Recommended IND
0	4	1
1	5	1
2	3	0
3	5	1
4	5	1

```
In [0]: #Let's plot and check the frequency of ratings
```

```
plt.figure(figsize=(12,6)) #set figure size
plt.hist(df_1['Rating']) #plot a histogram
plt.xlabel('Ratings') #set xlabel
plt.ylabel('counts') #set ylabel
plt.show()
```



We can see from the the distribution that most of the product are highly rated.Most are rated grater than 3.

```
In [0]: #crosstab to check the rating of the product and whether website recommends or not.
cross_tab = pd.crosstab(df_1['Rating'],df_1['Recommended IND']).T
cross_tab
```

Rating	1	2	3	4	5
Recommended IND					
0	826	1471	1682	168	25
1	16	94	1189	4909	13092

As we can see mostly high rated products are recommended by the website. We want statstical method to check whether the website's recommendation are dependent on ratings or not.

```
In [0]: #H0 : Recommended IND is independent of Ratings
#H1 : Recommended IND is not independent of Ratings
#Alpha : 0.05
```

```
In [0]: alpha = 0.05
```

```
stats,p_value,degrees_of_freedom,expected = chi2_contingency(cross_tab)
if p_value > alpha:
    print(f' Accept Null Hypothesis\n P-Value is {p_value}\n Recommendations are Independent of Ratings')
else:
    print(f' Reject Null Hypothesis\n P-Value is {p_value}\n Recommendations are not Independent of Ratings')
```

```
Reject Null Hypothesis
P-Value is 0.0
Recommendations are not Independent of Ratings
```

Chi2 independency test tells us that the Recommendations are not independent of ratings. We can also check it using

```
In [0]: recommended = df_1[df_1['Recommended IND']==1] #store all the recommended products in a recommended
not_recommended = df_1[df_1['Recommended IND']==0] #store all the not recommended products in a not recomm
```



```
In [0]: recommended['Rating'].value_counts() #check the value counts in recommended
```

```
5    13092
4     4909
3     1189
2         94
1         16
Name: Rating, dtype: int64
```

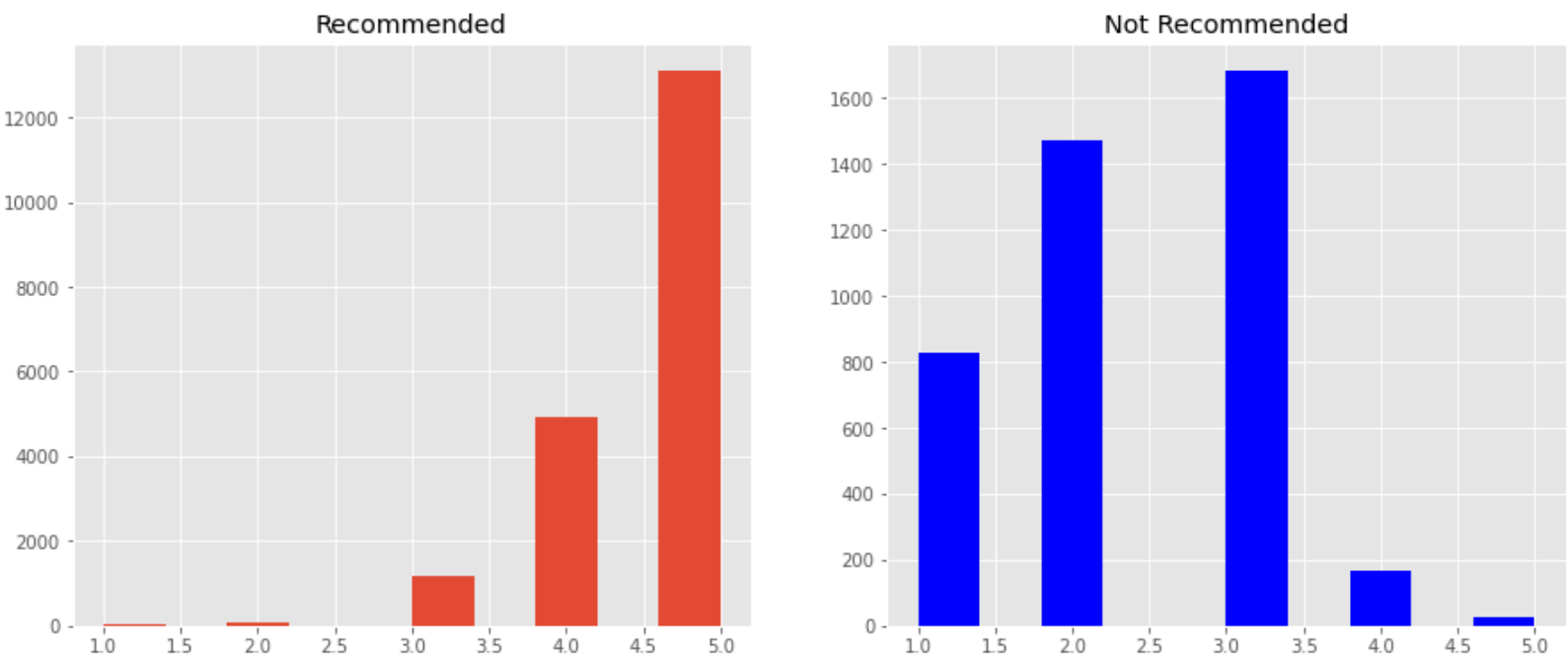
```
In [0]: not_recommended['Rating'].value_counts() #check the values count in not recommended
```

```
3     1682
2     1471
1      826
4      168
5       25
Name: Rating, dtype: int64
```

Most of the recommended products has higher ratings. And Those which were not recommended have lower rating. Indeed the website recommendation is dependent on ratings.

We can check it visually using histogram as well.

```
In [0]: #plot different histograms for recommended and not recommended ratings
fig,(ax1,ax2) = plt.subplots(1,2,figsize=(15,6))
recommended['Rating'].hist(ax=ax1)
ax1.set_title('Recommended')
not_recommended['Rating'].hist(ax=ax2,color='blue')
ax2.set_title('Not Recommended')
plt.show()
```



It is obvious from the plot that the Recommendation are dependent on Ratings. As we can see in graph the red bars shows that the recommended products has ratings greater than three. And the bars shows that the not recommended products were those which has lower ratings.

Hypothesis Testing Example (Blood Pressure Dataset)

```
In [0]: #install researchpy
!pip install researchpy

Collecting researchpy
  Downloading https://files.pythonhosted.org/packages/c2/e4/6fef21ad13c0b48ccba3ad8bd0bd65af0c9eb1aad9a82fec66c9de1ed/researchpy-0.1.9-py3-none-any.whl
Requirement already satisfied: statsmodels in /usr/local/lib/python3.6/dist-packages (from researchpy) (0.10.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from researchpy) (1.18.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.6/dist-packages (from researchpy) (1.4.1)
Requirement already satisfied: pandas in /usr/local/lib/python3.6/dist-packages (from researchpy) (1.0.3)
Requirement already satisfied: patsy>=0.4.0 in /usr/local/lib/python3.6/dist-packages (from statsmodels->researchpy) (0.5.1)
Requirement already satisfied: python-dateutil>=2.6.1 in /usr/local/lib/python3.6/dist-packages (from pandas->researchpy) (2.8.1)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.6/dist-packages (from pandas->researchpy) (2018.9)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from patsy>=0.4.0->statsmodels->researchpy) (1.12.0)
Installing collected packages: researchpy
Successfully installed researchpy-0.1.9
```

```
In [0]: #import the libraries
import statsmodels.api as sm
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib import style
style.use('ggplot')
from sklearn.preprocessing import scale
import researchpy as rp
import warnings
from scipy import stats
%matplotlib inline
```

```
In [0]: #read dataset
df = pd.read_csv('/content/drive/My Drive/data_set/blood_pressure.csv')
```

```
In [0]: #check shape
df.shape

(120, 5)
```

```
In [0]: df.head() #check the head
```

	patient	sex	agegrp	bp_before	bp_after
0	1	Male	30-45	143	153
1	2	Male	30-45	163	170
2	3	Male	30-45	153	168
3	4	Male	30-45	153	142
4	5	Male	30-45	146	141

```
In [0]: #check info
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 120 entries, 0 to 119
Data columns (total 5 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   patient     120 non-null   int64
1   sex         120 non-null   object
2   agegrp      120 non-null   object
3   bp_before   120 non-null   int64
4   bp_after    120 non-null   int64
dtypes: int64(3), object(2)
memory usage: 4.8+ KB
```

```
In [0]: #check null values
df.isnull().sum()
```

```
patient      0
sex           0
agegrp       0
bp_before    0
bp_after     0
dtype: int64
```

Pair sample t-test

Pair sample t-test is used to check the mean of two groups at different point in time. Here we want to know whether the bp of patients before and after taking the drug is same or difference. We want to know whether the drug has any effect on patients bp or not.

```
In [0]: #check the description of the data
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
patient	120.0	60.500000	34.785054	1.0	30.75	60.5	90.25	120.0
bp_before	120.0	156.450000	11.389845	138.0	147.00	154.5	164.00	185.0
bp_after	120.0	151.358333	14.177622	125.0	140.75	149.5	161.00	185.0

Here we can observe that the mean before intervention is 156.45 and mean after the intervention is 151.36. clearly we can see that there is difference in the mean before and after the intervention but is this difference statstically significant?

Check the assumptions

Check for equal variances

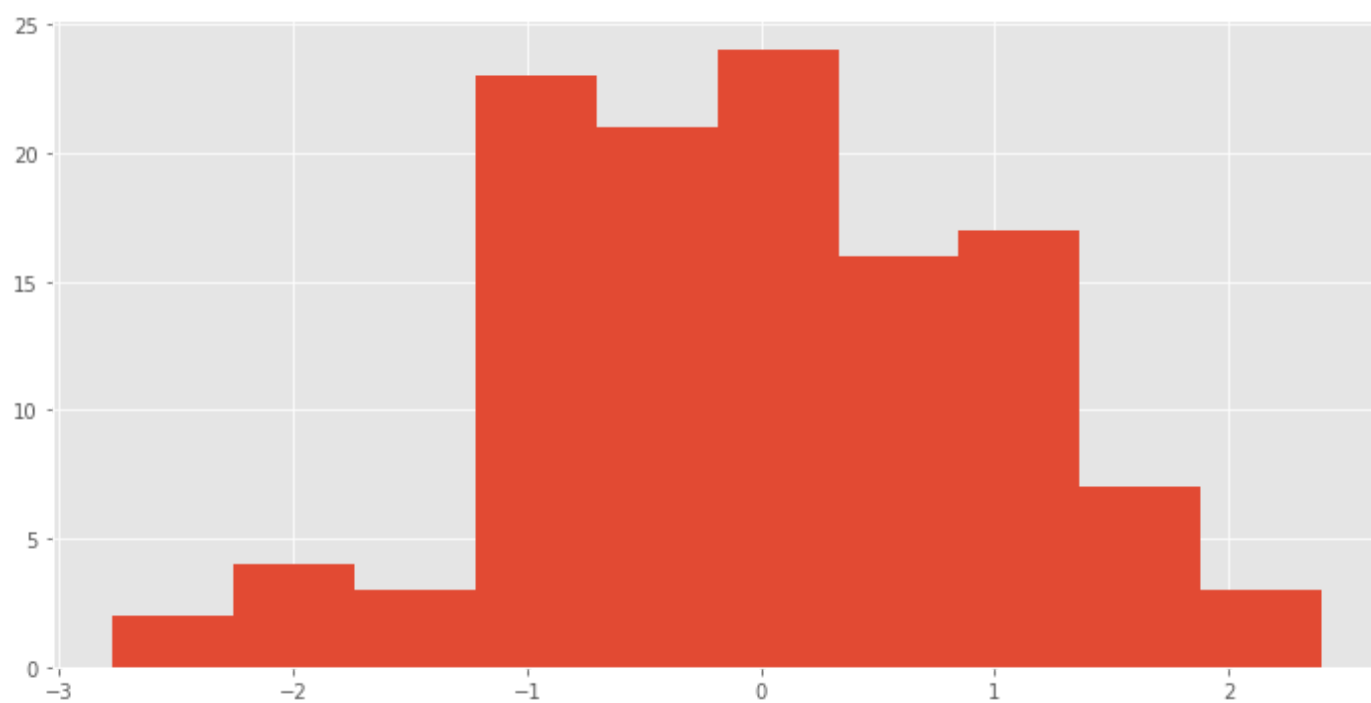
```
In [0]: #Use Levene's test to check variances
# H0 : Varinaces are same
# H1 : Variances are not same
#Alpha : 0.05

In [0]: alpha = 0.05
Stats,p_value = stats.levene(df['bp_before'],df['bp_after'])
if p_value > alpha :
    print('Failed to reject null hypothesis, variances are same')
else :
    print('Reject null hypothesis, Variances are not same')
```

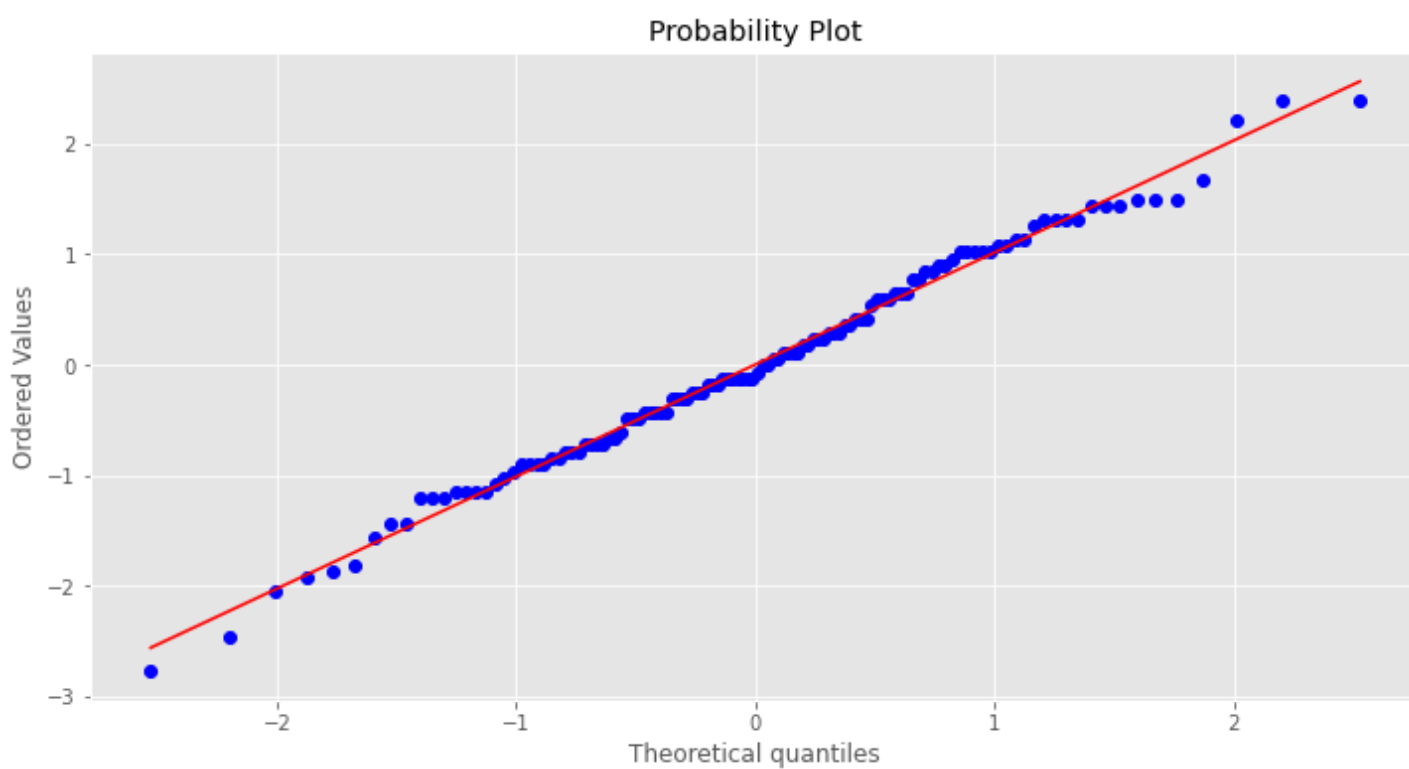
```
Reject null hypothesis, Variances are not same
```

Check for normality

```
In [0]: diff = scale(df['bp_before'] - df['bp_after']) #take the difference b/w before and after, and scale the differences
#plot histogram of scaled differences
plt.figure(figsize=(12,6))
plt.hist(diff)
plt.show()
```



```
In [0]: #q-q plot to check normality
plt.figure(figsize=(12,6))
stats.probplot(diff, plot=plt, dist = 'norm')
plt.show()
```



```
In [0]: #Let's use shapiro-wilk test to check normality
#H0 : Residuals are normally distributed
#H1 : Residuals are not normally distributed
alpha = 0.05
stats,p_value = stats.shapiro(diff)
if p_value > alpha:
    print('Failed to reject null hypothesis \n P-value: {}'.format(p_value))
else:
    print('Reject null hypothesis \n P-value: {}'.format(p_value))
```

```
Failed to reject null hypothesis
P-value: 0.7841646671295166
```

```
In [0]: #H0 : There's no difference in mean(No difference before and after an intervention)
#H1 : There's a difference in mean (There is difference before and after an intervention)
#Alpha : 0.5
alpha = 0.05
statistic , p_value = stats.ttest_rel(df['bp_before'],df['bp_after'])
if p_value > alpha:
    print(f'Fail to reject Null Hypothesis p-value is {p_value}')
else:
    print(f'Reject Null Hypothesis \nP_value : {p_value}')
```

Reject Null Hypothesis
P_value : 0.0011297914644840823

result shows that we can reject our null hypothesis and accept that there is a significant difference before and after intervention.

But while checking for assumptions of equal variance using leven's test ----- we saw that the variances are not same in both the sample. In such condition when the sample variances are not same we use Wilcoxon Signed Rank test.

```
In [0]: #we will use researchpy and see it's fusefullness here.
rp.ttest(df['bp_before'],df['bp_after'], equal_variances = False, paired =True)
```

	Wilcoxon signed-rank test	results
0	Mean for bp_before =	156.450000
1	Mean for bp_after =	151.358333
2	T value =	2234.500000
3	Z value =	-3.191600
4	Two sided p value =	0.001400
5	r =	-0.206000

Here we have used researchpy library for Wilcoxon Signed Rank test. We have specified equal_variances = False as the sample variances are not equal and paired=True for paired sample t-test.

We see that p-value (0.0014) is less than alpha (0.05). So we can reject null hypothesis.

Thank You