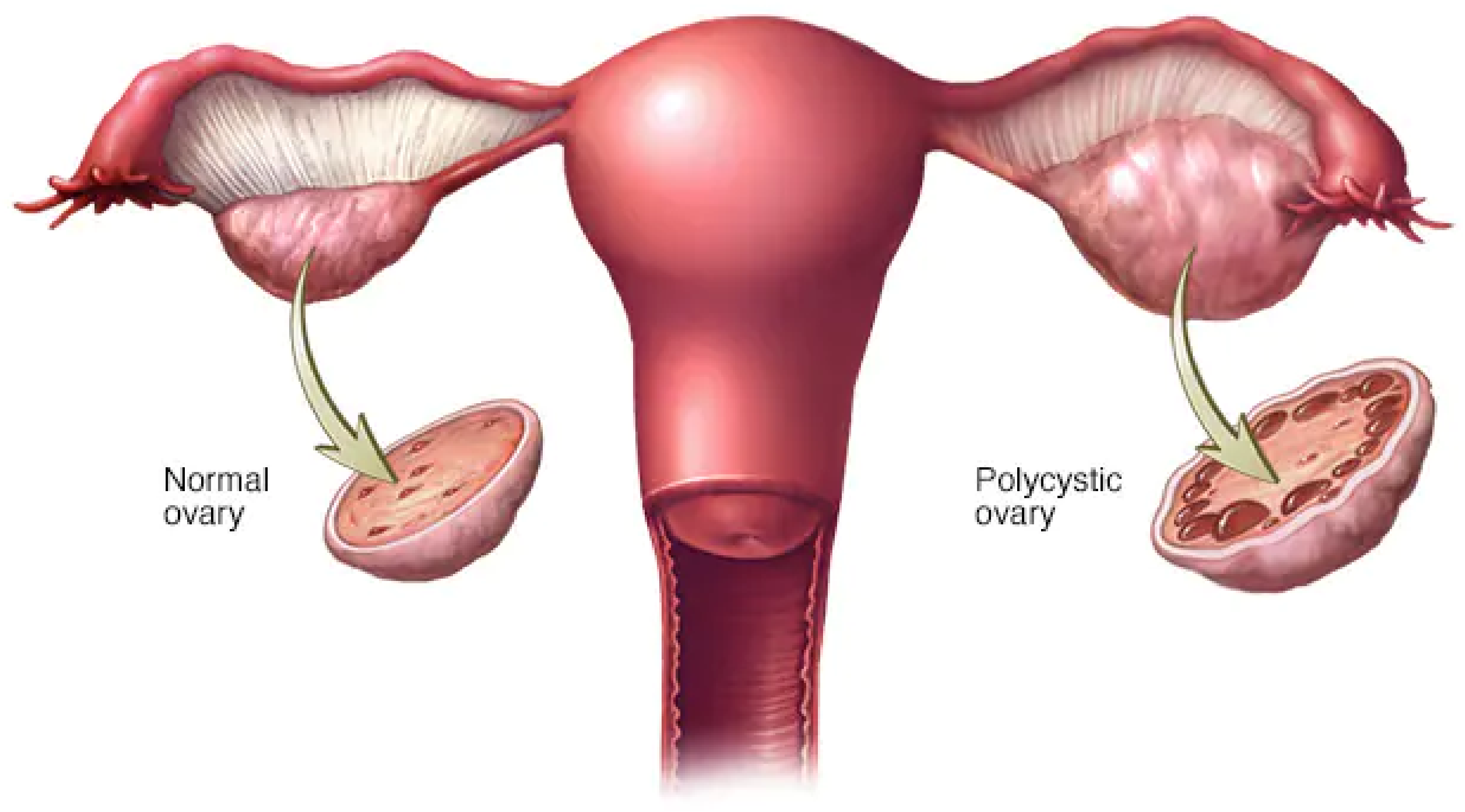
Case Study of Categorizing the Factors in women with PCOS

Table of Contents

- Abstract
- Chapters
  - Chp. 1 Introduction
  - Chp. 2 Methodology
  - Chp. 3 Data Description
  - Chp. 4 Analysis and Results
    - 4.1 Comparing Two Samples
    - 4.2 The Analysis of Variance
    - 4.3 The Analysis of Categorical Data
    - 4.4 Logistic Regression
    - 4.5 Resampling Methods
    - 4.6 Linear Model Selection and Regularization
    - 4.7 Moving Beyond Linearity
- Conclusion
- References

Abstract

This project focuses on the factors that affect an ovarian condition in most women called PolyCystic Ovarian Syndrome. Polycystic Ovary Syndrome (PCOS) is a health problem that affects women of childbearing age.



Women with PCOS may have infrequent or prolonged menstrual periods or excess male hormone (androgen) levels. The ovaries may develop numerous small collections of fluid (follicles) and fail to regularly release eggs. Women with PCOS have a hormonal imbalance and metabolism problems that may affect their overall health. PCOS manifests as a variety of symptoms.

Polycystic Ovary Syndrome (PCOS): Symptoms
SYMPTOMS VARY FROM WOMAN TO WOMAN
Infrequent, absent, and/or irregular menstrual periods
Male-pattern baldness or thinning hair
Acne, oily skin, or dandruff
Anxiety or depression
Cysts in the ovaries
Dark patches on skin
Weight gain
Infertility
Pelvic pain
Hirsutism
Sleep apnea , etc

The exact cause of PCOS is unknown. Early diagnosis and treatment and weight loss may reduce the risk of long-term complications such as type 2 diabetes and heart disease before they turn 40 yrs old.

PCOS affects 1-in-10 women:

> 10% women of child bearing age estimated to have PCOS
> 50% women go undiagnosed
> 4.3 Billion cost to the American healthcare system to diagnose and treat women with PCOS
> The risk of women with PCOS developing endometrial cancer is increased by 3 times

Thus, in this project we will be focusing on different factors that affect the menstruation cycle, the infertility and the well-being of any woman based on the health and lifestyle. My goal is to use the PCOS dataset on a variety of statistical methods that can be implemented to extract necessary features and get accurate predictions

1. Introduction

PCOS is a growing cause in women these days due to the changes in lifestyle. The motivation of this project is to implement various statistical methods which includes ANOVA, analysis of categorical data, different kinds of distributions, various regression models, resampling methods, etc. to analyse real data.

All above mentioned methods are used to analyse the PCOS dataset by comparing the samples in the dataset and then finding correlations between the important attributes of the dataset. Then, by applying various models, the accuracy of the different models can be checked and we can get the best fit for the dataset model by checking the accuracy. By implemeting Forward and Backward elimination, best features can be obtained for analysis using the components.

2. Methodology

---

1. Normality Test - First, we perform the normality test to check if the data is normal or not. For this we apply the multivariate normality test and get overall normality. We use the **multivariate_normality function** from *pingouin* library. Then, to check for each column separately we apply the Shapiro Wilk Test on each column individually and obtain the results. We import the **shapiro** method from the *scipy.stats* library. Based on the normality test, we perform the later analysis of ANOVA and categorical data.

---

1. Mann-Whitney Test - Since, the data is not normal, Mann-whitney test is applied for the comparison of two samples. Here, I applied the test between the response variable *data["PCOS(Y/N)"]* and remaining columns of the dataset. For this, we use the **mannwhitneyu** method from the *scipy.stats* library.

---

1. Kruskal Wallis Test - Again, since the data is not normal, we apply the Kruskal Wallis test for One-way ANOVA. For this, we use the **kruskal** method from the *scipy.stats* library. From this we understand if the means are different are not. But we do not understand for which samples they are actually different.

---

1. Nemenyi Post-hoc Test - To check which samples are different from each other, we apply the Nemenyi Test. We check the test for the same columns that we used in the Kruskal Wallis test, to get insight on which columns have different means. For this, we use the **posthoc_nemenyi_friedman** method from _scikit*posthocs* library.

---

1. Chi-square test for analysis - For analysis of categorical data, we use chi-square test as we have more than two columns to analyse. Using the chi-square test, we get the p-value for testing the hypothesis. If the p-value is significantly lower, we use the Cramver's V to conclude our interpretation. For this, we use the **crosstab** method from *researchpy* library.

---

1. Logistic Regression - Since my response variable is binary, I used the logistic regression to analyse the accuracy. For this I used the **LogisticRegression** method from _sklearn.linear*model*. The classification report shows the accuracy and the F-1 score.

---

1. Ridge and Lasso Regression - After performing logistic regression, I perform Ridge regression and Lasso regression for the logistic one to check the accuracy of the same. For this, I used **Ridge()** and **Lasso()** methods respectively from _sklearn.linear*model*.

---

1. Forward and Backward Selection - We then perform forward and backward selection in order to get the top best features i.e. attributes from the dataset.

---

1. Bootstrap and k-fold validation - We perform Bootstraping and k-fold validation amongst the resampling method using Lasso and compare the accuracy of the same.

---

1. Principle Component Analysis (PCA) - Using the best features that we obtained, we apply the technique of dimensionality reduction and obtain the confusion matrix which helps us obtain the accuracy and compare with the rest. For this, we import **PCA** method from *sklearn.decomposition* library.

---

1. Polynomial Regression - Since, fitting data in a linear manner can give far less accuracy, we use polynomial regression. We use **PolynomialFeatures** method from *sklearn.preprocessing* library and compare the errors for second, third, fourth and fifth order polynomials.

---

1. Generalised Additive Models (GAM) - In GAM, the linear response variable "PCOS(Y/N" depends linearly on unknown smooth functions of some predictor variables from the dataset. We infer the performance based on them.

3. Data Description

This data is collected from 10 different hospitals across Kerala, India. This dataset contains some physical and clinical parameters to determine PCOS and infertility related issues. There are a total of 16 parameters we have taken into consideration where our response variable is "PCOS (Y/N" and rest are predictors.

1. PCOS (Y/N) - PCOS presence in the patient (Y means positive and N means negative results)
2. Age (yrs) - Age of the patient
3. BMI - Body mass index of the patient
4. Waist:Hip Ratio - Waist to hip ratio of the patient
5. Cycle(R/I) - Menstrual Cycle regurality (R mean Regular and I means Irregular)
6. Cycle length(days) - Cycle length of the patient (in terms of number of days)
7. Marraige Status (Yrs) - No. of year the patient has been married
8. I beta-HCG(mIU/mL) - Quantitative hCG Blood Pregnancy Test 1
9. II beta-HCG(mIU/mL) - Quantitative hCG Blood Pregnancy Test 2
10. AMH(ng/mL) - Anti-Müllerian Hormone content in the patient (in ng/mL)
11. Hb(g/dl) - Hemoglobin of the patient (in g/dl)
12. BP Diastolic (mmHg) - Blood Pressure of the patient
13. Vit D3 (ng/mL) - Vitamin D3 content in the patient to check for deficieny
14. Follicle No. (L) - Follicle number for the left ovary
15. Follicle No. (R) - Follicle number for the right ovary
16. Endometrium (mm) - Thickness of the Endometrium layer (in mm)

Importing the required libraries

In [29]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as py

import scipy.stats as stats

from pingouin import multivariate_normality

import scipy.stats
from scipy.stats import shapiro
from scipy.stats import lognorm
from scipy.stats import mannwhitneyu
from scipy.stats import kruskal

import scikit_posthocs as sp

import researchpy as rp

import statsmodels.api as sm

from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import Ridge
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error as mse
from sklearn.metrics import mean_absolute_error as mae
from sklearn.feature_selection import SelectKBest,f_classif
from sklearn.model_selection import train_test_split
```

```
import math

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

Displaying the dataset

In [30]:

```
data = pd.read_csv('PCOS_data.csv',
                   usecols = ['PCOS (Y/N)',
                              ' Age (yrs)',
                              'BMI',
                              'Waist:Hip Ratio' ,
                              'Cycle(R/I)',
                              'Cycle length(days)',
                              'Marraige Status (Yrs)',
                              'I beta-HCG(mIU/mL)',
                              'II beta-HCG(mIU/mL)',
                              'AMH(ng/mL)',
                              'Hb(g/dl)',
                              'BP _Diastolic (mmHg)',
                              'Vit D3 (ng/mL)',
                              'Follicle No. (L)',
                              'Follicle No. (R)',
                              'Endometrium (mm)'])
data
```

Out[30]:

	PCOS (Y/N)	Age (yrs)	BMI	Hb(g/dl)	Cycle(R/I)	Cycle length(days)	Marraige Status (Yrs)	I beta-HCG(mIU/mL)	II beta-HCG(mIU/mL)	Waist:Hip Ratio	AMH(ng/mL)	Vit D3 (ng/mL)	BP _Diastolic (mmHg)	Follicle No. (L)
0	0	28	19.3	10.48	2	5	7.0	1.99	1.99	0.83	2.07	17.1	80	3
1	0	36	24.9	11.70	2	5	11.0	60.80	1.99	0.84	1.53	61.3	70	3
2	1	33	25.3	11.80	2	5	10.0	494.08	494.08	0.90	6.63	49.7	80	13
3	0	37	29.7	12.00	2	5	4.0	1.99	1.99	0.86	1.22	33.4	70	2
4	0	25	20.1	10.00	2	5	1.0	801.45	801.45	0.81	2.26	43.8	80	3
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
536	0	35	18.5	11.00	2	5	8.0	1.99	1.99	0.93	1.7	36.6	70	1
537	0	30	25.3	10.80	2	5	4.0	80.13	1.99	0.94	5.6	23.0	70	9
538	0	36	23.4	10.80	2	6	8.0	1.99	1.99	0.93	3.7	22.5	80	1
539	0	27	22.2	12.00	4	2	2.0	292.92	1.99	0.93	5.2	22.4	70	7
540	1	23	30.1	10.20	4	7	2.0	1.99	1.99	0.96	20	17.4	70	9

541 rows × 16 columns

Dropping the missing values from the dataset

In [31]:

```
# Checking for Nan values
data.isna().sum()
```

Out[31]:

```
PCOS (Y/N)                0
 Age (yrs)                0
BMI                       0
Hb(g/dl)                  0
Cycle(R/I)                0
Cycle length(days)        0
Marraige Status (Yrs)     1
I beta-HCG(mIU/mL)        0
II beta-HCG(mIU/mL)       0
Waist:Hip Ratio           0
AMH(ng/mL)                0
Vit D3 (ng/mL)            0
BP _Diastolic (mmHg)      0
Follicle No. (L)          0
Follicle No. (R)          0
Endometrium (mm)          0
dtype: int64
```

After checking, we found a missing value entry in the "Marraige Status (Yrs)" column. Thus, we need to drop it as below.

In [32]:

```
data = data.apply(pd.to_numeric, errors = 'coerce')
data = data.dropna()
data.isna().sum()
```

Out[32]:

```
PCOS (Y/N)                0
 Age (yrs)                0
BMI                       0
Hb(g/dl)                  0
Cycle(R/I)                0
Cycle length(days)        0
Marraige Status (Yrs)     0
I beta-HCG(mIU/mL)        0
II beta-HCG(mIU/mL)       0
```

```
Waist:Hip Ratio          0
AMH(ng/mL)               0
Vit D3 (ng/mL)           0
BP _Diastolic (mmHg)     0
Follicle No. (L)         0
Follicle No. (R)         0
Endometrium (mm)         0
dtype: int64
```

After dropping, we reassure if there are any missing values remaining.

To understand the dataset and its attributes better, we perform some **Exploratory Data Analysis**.

Exploratory Data Analysis

At first, we describe the data and get the summary of its columns.

In [33]:

```python
data.describe(include = 'all')
```

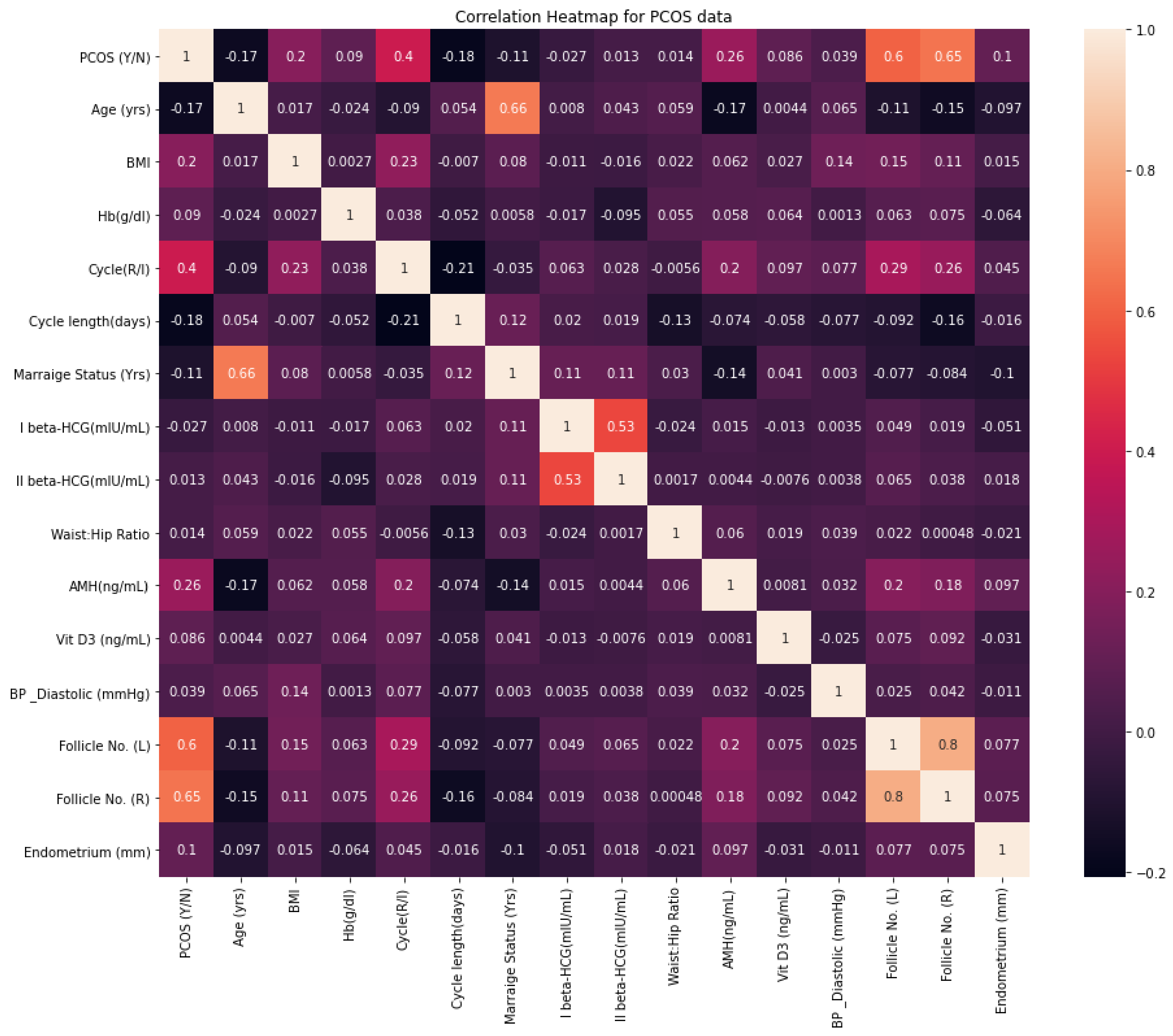
Out[33]:

	PCOS (Y/N)	Age (yrs)	BMI	Hb(g/dl)	Cycle(R/I)	Cycle length(days)	Marraige Status (Yrs)	I beta-HCG(mIU/mL)	II beta-HCG(mIU/mL)	Waist:Hip Ratio	AMH(ng/mL)	(
count	538.000000	538.000000	538.000000	538.000000	538.000000	538.000000	538.000000	538.000000	538.000000	538.000000	538.000000	538
mean	0.325279	31.427509	24.318401	11.160000	2.559480	4.936803	7.689033	667.215271	239.550333	0.891543	5.596028	50
std	0.468915	5.404955	4.056571	0.868844	0.901681	1.493531	4.806181	3358.025666	1608.201951	0.046071	5.860493	347
min	0.000000	20.000000	12.400000	8.500000	2.000000	0.000000	0.000000	1.300000	0.990000	0.760000	0.100000	0
25%	0.000000	28.000000	21.625000	10.500000	2.000000	4.000000	4.000000	1.990000	1.990000	0.860000	2.010000	20
50%	0.000000	31.000000	24.200000	11.000000	2.000000	5.000000	7.000000	19.375000	1.990000	0.890000	3.700000	25
75%	1.000000	35.000000	26.675000	11.775000	4.000000	5.000000	10.000000	297.050000	99.175000	0.930000	6.890000	34
max	1.000000	48.000000	38.900000	14.800000	5.000000	12.000000	30.000000	32460.970000	25000.000000	0.980000	66.000000	6014

Then, I tried to obtain the correlation of the variable columns and displayed the same using a *heatmap*.

In [34]:

```python
# Correlation heatmap

py.figure(figsize=(15,12))
heatmap = sns.heatmap(data.corr(),annot=True)
heatmap.set_title('Correlation Heatmap for PCOS data');
```



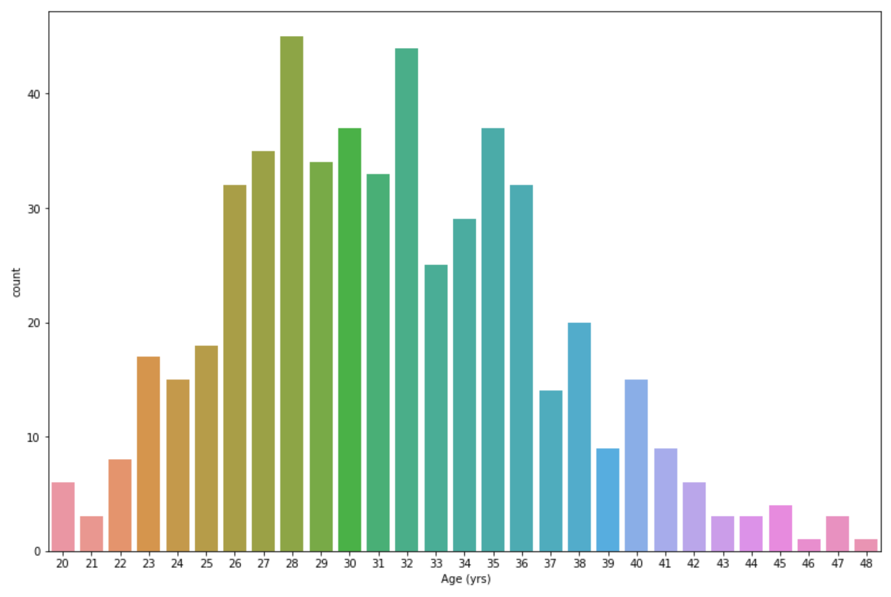
The above plot gave an insight into each of the columns and their correlation with each of the attributes.

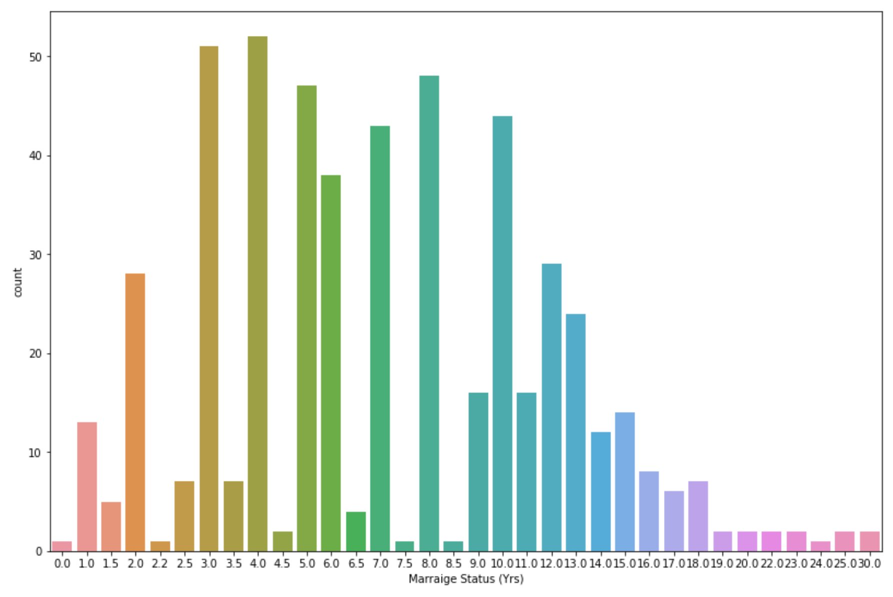
This encorgaed me to plot countplots for

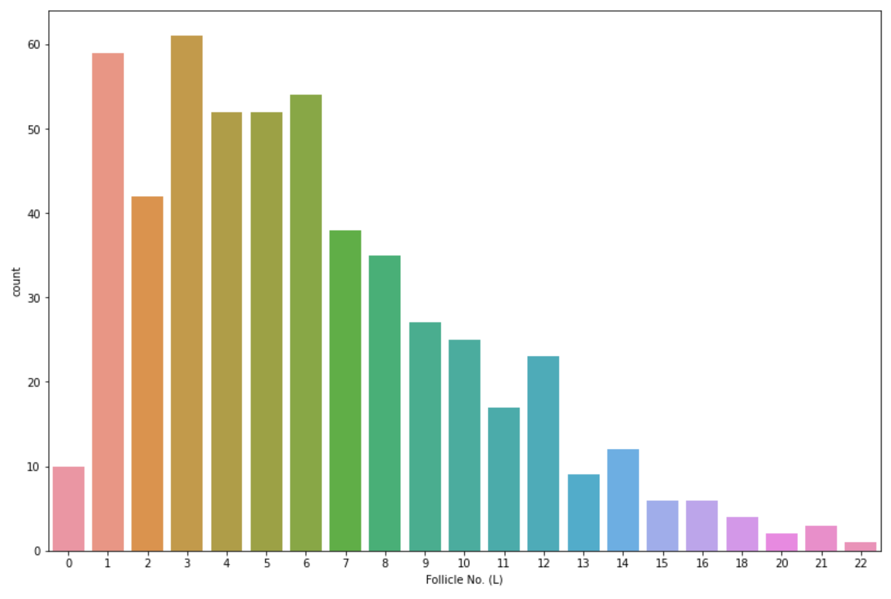
> - The age of the patient - to know exactly at what age is PCOS much more frequent
> - The follicle no of left ovary - to know the number of the follicles in the left ovary of the patient with PCOS
> - The follicle no of right ovary - to know the number of the follicles in the right ovary of the patient with PCOS
> - The marriage status - to know if by the number of years of marriage and intimacy with partner affect the the PCOS then how many years can affect the same

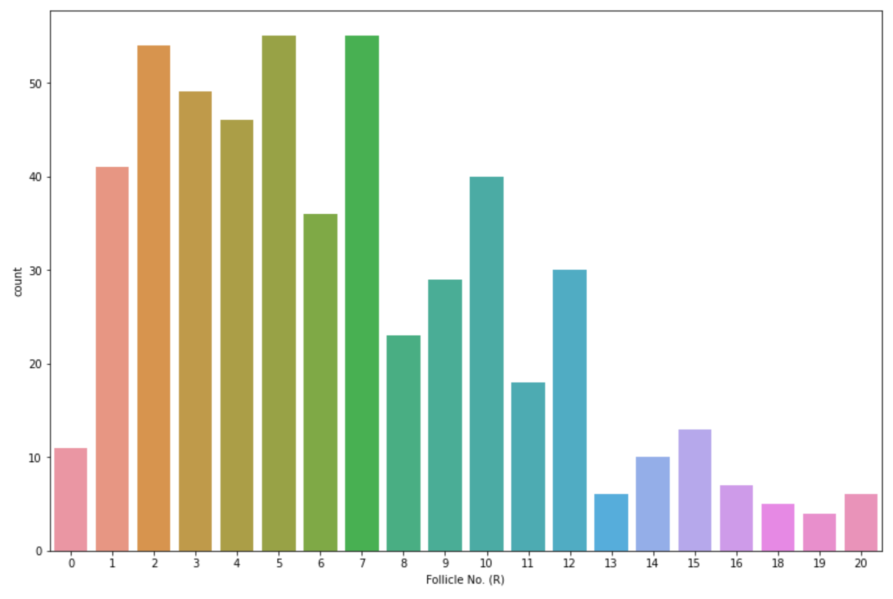
In [61]:

```
fig = py.figure(figsize=(30, 20))

py.subplot(2,2,1)
sns.countplot(data[" Age (yrs)"]);

py.subplot(2,2,2)
sns.countplot(data["Marraige Status (Yrs)"]);

py.subplot(2,2,3)
sns.countplot(data["Follicle No. (L)"]);

py.subplot(2,2,4)
sns.countplot(data["Follicle No. (R)"]);
```





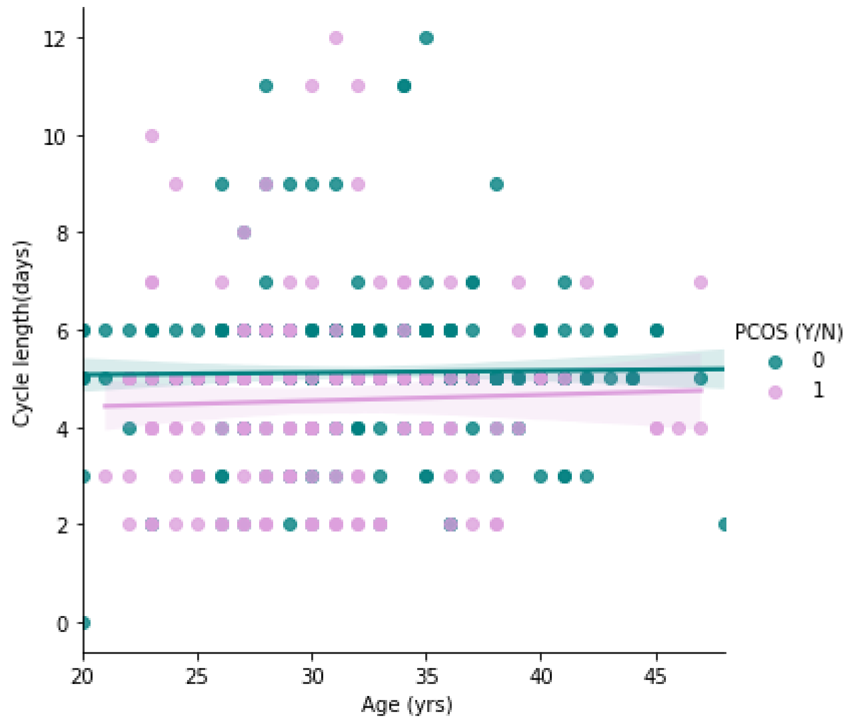




Since, certain aspects of the dataset were now clearer and the features were highlighted, I plotted the regression model fits for various such attributes against the response variable 'PCOS' of the dataset.

Patterns Of Length Of The Menstrual Cycle

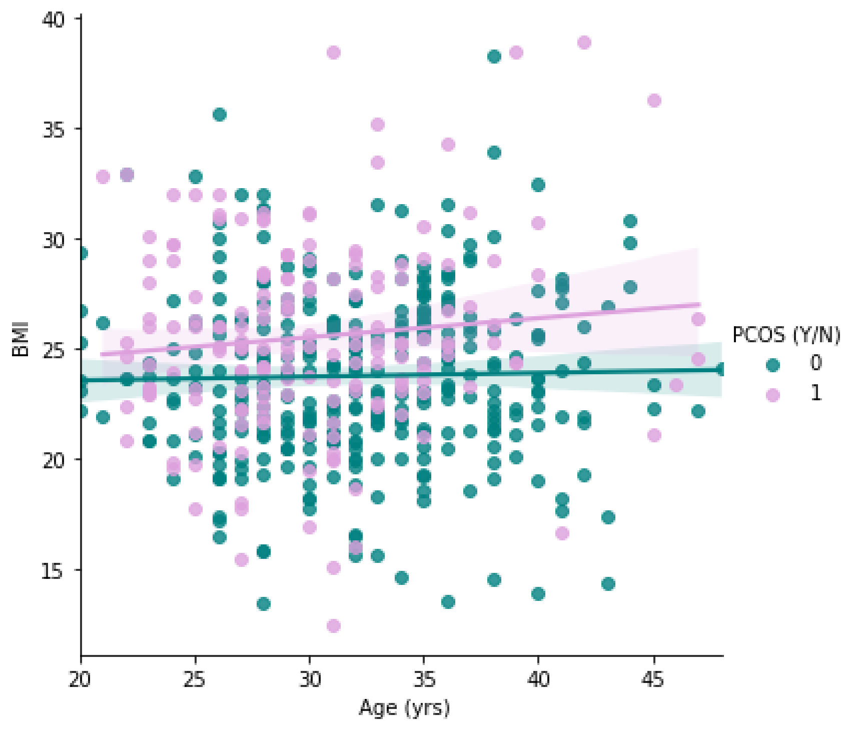
In [47]:
```
# Length of menstrual phase in PCOS vs normal

color = ["teal", "plum"]
fig = sns.lmplot(data=data,x=" Age (yrs)",y="Cycle length(days)", hue="PCOS (Y/N)",palette=color)
py.show(fig)
```



The length of the menstrual phase is overall consistent over different ages for normal cases. Whereas in the case of PCOD the length increased with age.

Patterns of BMI

In [48]:
```
# Pattern of weight gain (BMI) over years in PCOS and Normal.

fig= sns.lmplot(data =data,x=" Age (yrs)",y="BMI", hue="PCOS (Y/N)", palette= color )
py.show(fig)
```

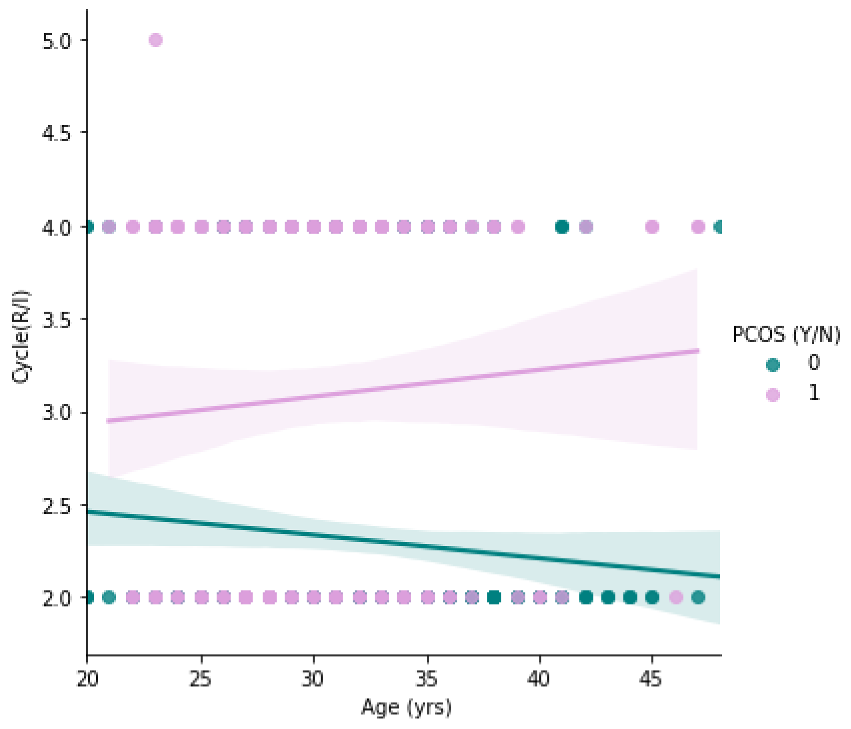


Body mass index (BMI) is showing consistency for normal cases. Whereas for PCOS the BMI increases with age.

Patterns Of Irregularity In Mensuration

> 4 indicates irregular menstrual cycle
> 2 indicates a regular menstrual cycle

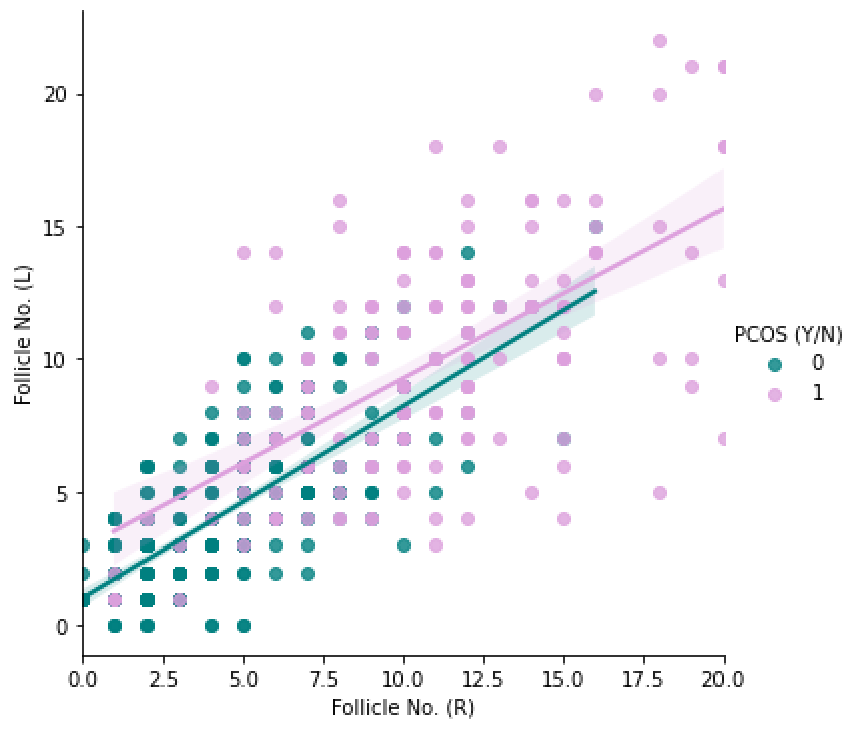
In [49]:
```
# Cycle irregularity with respect to age

sns.lmplot(data =data,x=" Age (yrs)",y="Cycle(R/I)", hue="PCOS (Y/N)",palette=color)
py.show()
```



The menstrual cycle becomes more regular for normal cases with age. Whereas, for PCOS the irregularity increases with age.

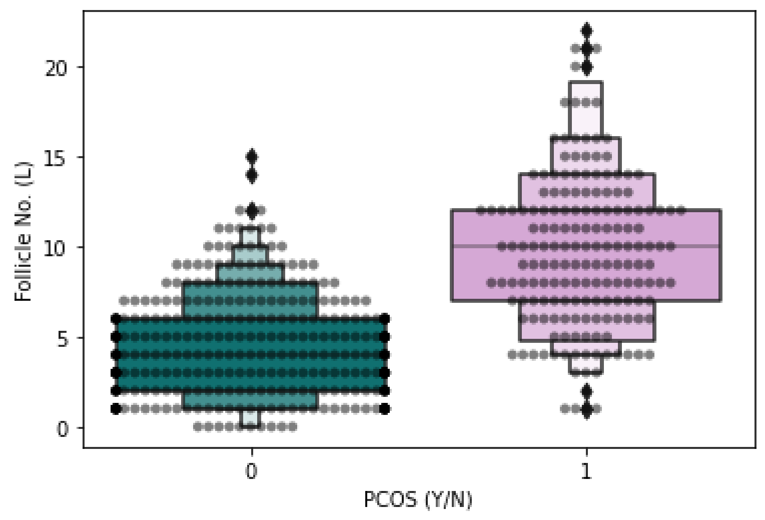
Patterns considering Number Of Follicles in both the left and right ovaries

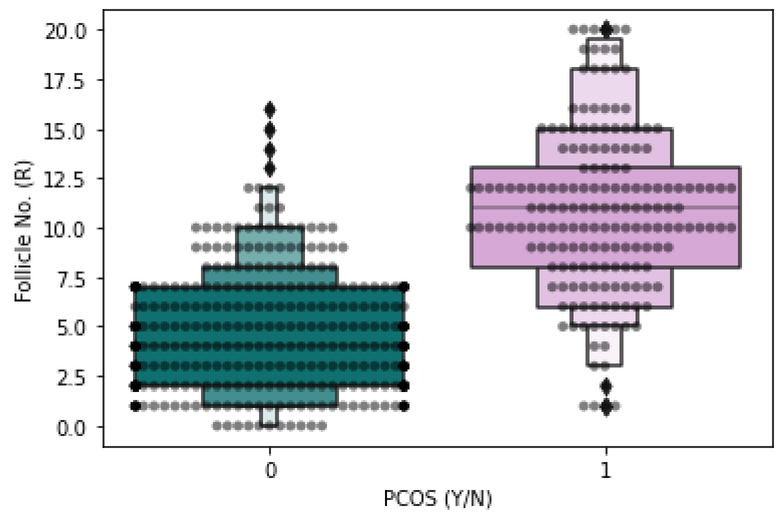
In [50]:
```
# Distribution of follicles in both ovaries.

sns.lmplot(data =data,x='Follicle No. (R)',y='Follicle No. (L)', hue="PCOS (Y/N)",palette=color)
py.show()
```



The distribution of follicles in both ovaries Left and Right are not equal for women with PCOS in comparison with the "Normal" patient. This peaked curiousty in exloring the same with boxen plot.

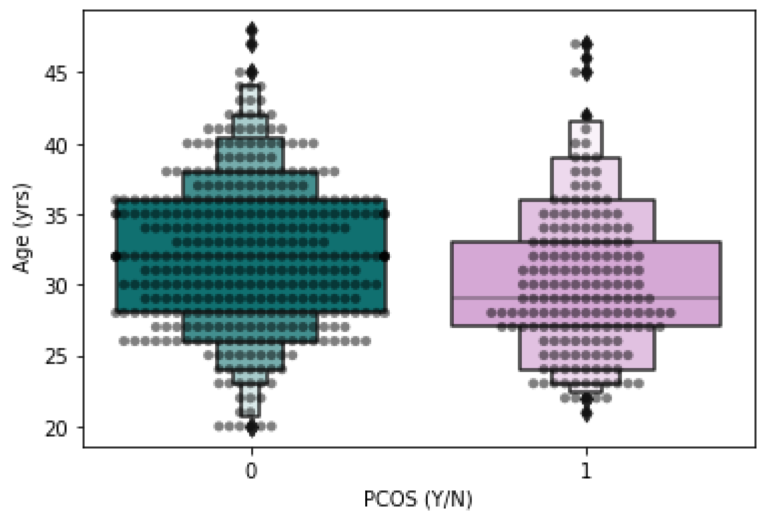
In [51]:
```
features = ["Follicle No. (L)","Follicle No. (R)"]
for i in features:
    sns.swarmplot(x=data["PCOS (Y/N)"], y=data[i], color="black", alpha=0.5 )
    sns.boxenplot(x=data["PCOS (Y/N)"], y=data[i], palette=color)
    py.show()
```

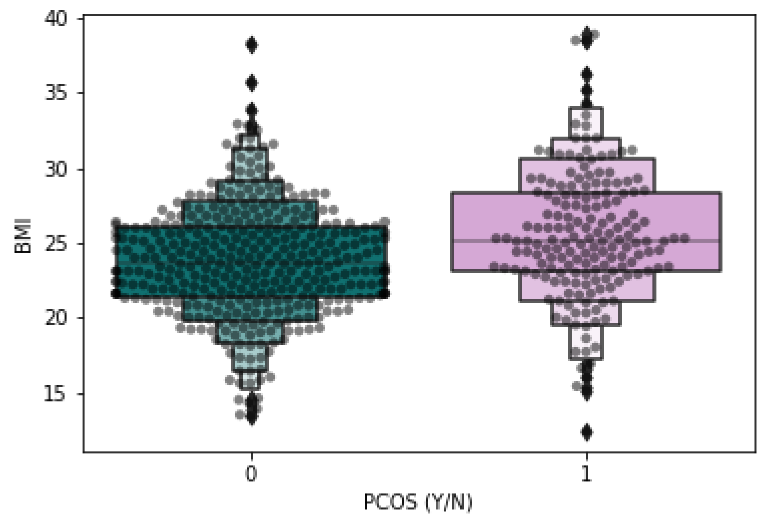


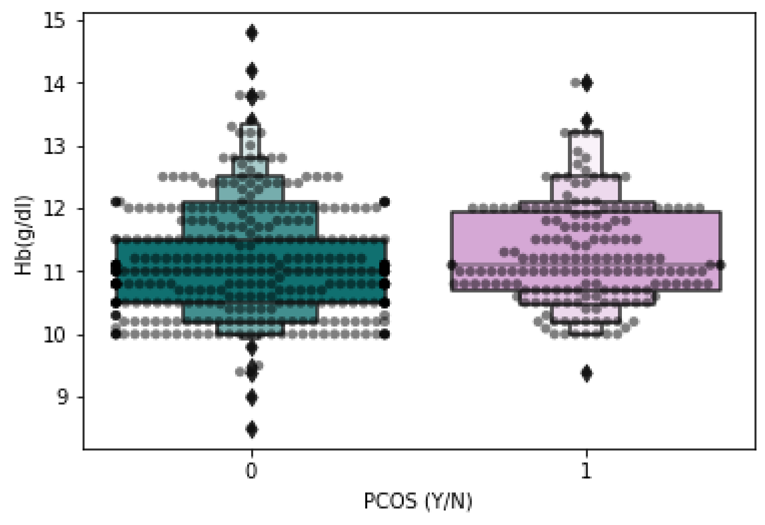


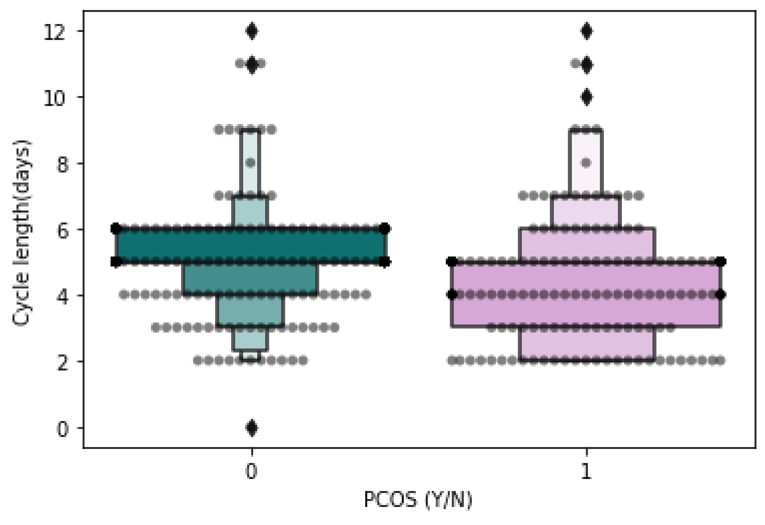
What we found out was that the number of follicles in women with PCOS is higher, as expected and are unequal as well.

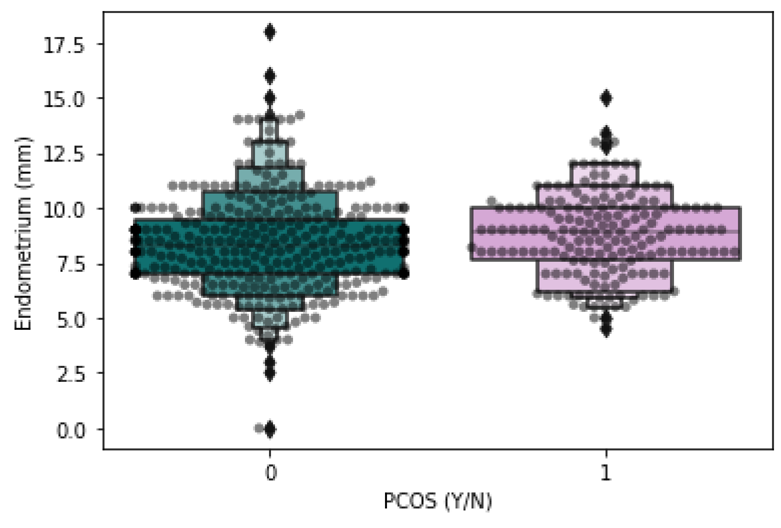
In [126…
```
# Miscellaneous EDA to check a few more columns as well.

features = [" Age (yrs)", "BMI", "Hb(g/dl)", "Cycle length(days)","Endometrium (mm)" ]
for i in features:
    sns.swarmplot(x=data["PCOS (Y/N)"], y=data[i], color="black", alpha=0.5 )
    sns.boxenplot(x=data["PCOS (Y/N)"], y=data[i], palette=color)
    py.show()
```











4. Analysis and Results

After performing the EDA, the features that our much better to provide better insights have become clearer.
It is now time to perform the statistical methods to predict the presence of PCOS and what factors affect the same.

For this, we first we need to check if our data is distributed normal or not.

Normality Test

To check if our data is normally distributed or not we perform the normality test. Since, we have many predictors, I performed the multivariate normality test on the PCOS dataset.

In [103…
```
test_df = data
multivariate_normality(test_df, alpha=.05)
```

Out[103…
```
HZResults(hz=1.7631692583597318, pval=0.0, normal=False)
```

We thus found that for overall dataset, the distribution came as **NOT NORMAL** as the results read normal as False.

Now, let us check the normality for each column using the **Shapiro Wilk Test**.

In [107…
```
# Checking Normality for each column
for i in data.columns:
    stat, p = shapiro(data[i])
    if p > 0.05:
        print("For column",i,"------------> Normal dist")
    else:
        print("For column",i,"-----------> Non-normal dist")
```

```
For column PCOS (Y/N) -----------> Non-normal dist
For column  Age (yrs) -----------> Non-normal dist
For column BMI -----------> Non-normal dist
For column Hb(g/dl) -----------> Non-normal dist
For column Cycle(R/I) -----------> Non-normal dist
For column Cycle length(days) -----------> Non-normal dist
For column Marraige Status (Yrs) -----------> Non-normal dist
For column I beta-HCG(mIU/mL) -----------> Non-normal dist
For column II beta-HCG(mIU/mL) -----------> Non-normal dist
For column Waist:Hip Ratio -----------> Non-normal dist
For column AMH(ng/mL) -----------> Non-normal dist
For column Vit D3 (ng/mL) -----------> Non-normal dist
For column BP _Diastolic (mmHg) -----------> Non-normal dist
For column Follicle No. (L) -----------> Non-normal dist
For column Follicle No. (R) -----------> Non-normal dist
For column Endometrium (mm) -----------> Non-normal dist
```

Here, we can find that for each column, the distribution is not normal.

Hence, we can apply the **non-parametric tests** on our dataset.

4.1 Comparing Two Samples

We found out that the data that we have is not normal. Hence, I apply the Mann-whitney test for non-parametric data to compare each column of my dataset with the response column and check if there is difference in them or not. Mann-Whitney test, also called as Wilcoxon rank sum test, is a non parametric test that allows two groups to be compared without making the assumption that values are following any specific distribution.

We consider the level of significance, alpha = 0.05

Thus, let the hypothesis be:

H_0 : There is no significant difference between the attributes

vs

H_1 : There is no significant difference between the attributes

In [129…
```
# Mann-Whitney Test
print("Therefore, following is the outcome:\n")
for i in data.columns:

    print("For column: ",i)

    stats, pvalue = mannwhitneyu(data["PCOS (Y/N)"], data[i], alternative='two-sided')
    print('Statistics=%.5f, p=%.5f' % (stats, pvalue))

    alpha = 0.05

    # conclusion
    if pvalue < alpha:
        print('Reject Null Hypothesis\nThus, There is significant difference between the factors')
    else:
        print('Do not Reject Null Hypothesis\nThus, There is no significant difference between the factors')
    print("----------------------------------------------------------------------------------\n")
```

```
Therefore, following is the outcome:

For column:  PCOS (Y/N)
Statistics=144722.00000, p=1.00000
Do not Reject Null Hypothesis
Thus, There is no significant difference between the factors
----------------------------------------------------------------------------------
```

```
For column:   Age (yrs)
Statistics=0.00000, p=0.00000
Reject Null Hypothesis
Thus, There is significant difference between the factors
---------------------------------------------------------------------------------

For column:  BMI
Statistics=0.00000, p=0.00000
Reject Null Hypothesis
Thus, There is significant difference between the factors
---------------------------------------------------------------------------------

For column:  Hb(g/dl)
Statistics=0.00000, p=0.00000
Reject Null Hypothesis
Thus, There is significant difference between the factors
---------------------------------------------------------------------------------

For column:  Cycle(R/I)
Statistics=0.00000, p=0.00000
Reject Null Hypothesis
Thus, There is significant difference between the factors
---------------------------------------------------------------------------------

For column:  Cycle length(days)
Statistics=356.50000, p=0.00000
Reject Null Hypothesis
Thus, There is significant difference between the factors
---------------------------------------------------------------------------------

For column:  Marraige Status (Yrs)
Statistics=1494.00000, p=0.00000
Reject Null Hypothesis
Thus, There is significant difference between the factors
---------------------------------------------------------------------------------

For column:  I beta-HCG(mIU/mL)
Statistics=0.00000, p=0.00000
Reject Null Hypothesis
Thus, There is significant difference between the factors
---------------------------------------------------------------------------------

For column:  II beta-HCG(mIU/mL)
Statistics=175.00000, p=0.00000
Reject Null Hypothesis
Thus, There is significant difference between the factors
---------------------------------------------------------------------------------

For column:  Waist:Hip Ratio
Statistics=94150.00000, p=0.00000
Reject Null Hypothesis
Thus, There is significant difference between the factors
---------------------------------------------------------------------------------

For column:  AMH(ng/mL)
Statistics=9362.50000, p=0.00000
Reject Null Hypothesis
Thus, There is significant difference between the factors
---------------------------------------------------------------------------------

For column:  Vit D3 (ng/mL)
Statistics=356.50000, p=0.00000
Reject Null Hypothesis
Thus, There is significant difference between the factors
---------------------------------------------------------------------------------

For column:  BP _Diastolic (mmHg)
Statistics=0.00000, p=0.00000
Reject Null Hypothesis
Thus, There is significant difference between the factors
---------------------------------------------------------------------------------

For column:  Follicle No. (L)
Statistics=8727.50000, p=0.00000
Reject Null Hypothesis
Thus, There is significant difference between the factors
---------------------------------------------------------------------------------

For column:  Follicle No. (R)
Statistics=7509.00000, p=0.00000
Reject Null Hypothesis
Thus, There is significant difference between the factors
---------------------------------------------------------------------------------

For column:  Endometrium (mm)
Statistics=713.00000, p=0.00000
Reject Null Hypothesis
Thus, There is significant difference between the factors
---------------------------------------------------------------------------------
```

Here, we can find that there is significant difference between the different columns of the dataset with the response variable.

4.2 The Analysis of Variance

Since, our data is non-parametric, for one-way ANOVA, I applied the **Kruskal Wallis test**.

To apply this test following assumptions are made:

> All samples are independent and randomly selected from population.
> Homogeneity of the variance of the population.
> The factor (indep. variable) is a categorical variable with two or more groups (treatments or levels).
> he response (dependent) variable is a numerical variable.

Here, let the hypothesis be:

H_0 : The means are same.

vs

H_1 : The means are different.

In [56]:

```
# Kruskal-Wallis Test
stats, pvalue = kruskal(data["PCOS (Y/N)"],
                        data[" Age (yrs)"],
                        data["BMI"],
                        data['Waist:Hip Ratio'] ,
                        data["Cycle(R/I)"],
                        data["I beta-HCG(mIU/mL)"],
                        data["II beta-HCG(mIU/mL)"],
                        data['BP _Diastolic (mmHg)'],
                        data['Hb(g/dl)'],
                        data['Follicle No. (L)'],
                        data['Follicle No. (R)'],
                        data['Endometrium (mm)'])

print('Statistics=%.5f, p=%.5f' % (stats, pvalue))
alpha = 0.05

if pvalue < alpha:
    print('Reject Null Hypothesis\nThus, the means are different for above columns.')
else:
    print('Do not Reject Null Hypothesis\nThus, the means are same for above columns.')
print("-----------------------------------------------------------------------------------\n")
```

```
Statistics=4540.11642, p=0.00000
Reject Null Hypothesis
Thus, the means are different for above columns.
-----------------------------------------------------------------------------------
```

After applying the test for the important features of the dataset such as Age, BMI, blood pressure, etc. we find out that the means for above features are different. But this is a collective result. We want to find out for which columns exactly are the means different.

For this reason, we apply the **Nemenyi posthoc test** for the above mentioned columns.

In [131…

```
# Nemenyi posthoc Test

df_f = np.array([data[" Age (yrs)"],
                 data["BMI"],
                 data['Waist:Hip Ratio'] ,
                 data["Cycle(R/I)"],
                 data["I beta-HCG(mIU/mL)"],
                 data["II beta-HCG(mIU/mL)"],
                 data['BP _Diastolic (mmHg)'],
                 data['Hb(g/dl)'],
                 data['Follicle No. (L)'],
                 data['Follicle No. (R)'],
                 data['Endometrium (mm)']])

# Conduct the Nemenyi post-hoc test
a = sp.posthoc_nemenyi_friedman(df_f.T)
print("Thus, the Nemeyni test result is\n",a,"\n")
```

```
Thus, the Nemeyni test result is
            0         1      2      3      4         5      6      7         8   \
0   1.000000  0.011104  0.001  0.001  0.001  0.001000  0.001  0.001  0.001000
1   0.011104  1.000000  0.001  0.001  0.001  0.001000  0.001  0.001  0.001000
2   0.001000  0.001000  1.000  0.001  0.001  0.001000  0.001  0.001  0.001000
3   0.001000  0.001000  0.001  1.000  0.001  0.001000  0.001  0.001  0.001000
4   0.001000  0.001000  0.001  0.001  1.000  0.001000  0.001  0.900  0.001000
5   0.001000  0.001000  0.001  0.001  0.001  1.000000  0.001  0.001  0.021921
6   0.001000  0.001000  0.001  0.001  0.001  0.001000  1.000  0.001  0.001000
7   0.001000  0.001000  0.001  0.001  0.900  0.001000  0.001  1.000  0.001000
8   0.001000  0.001000  0.001  0.001  0.001  0.021921  0.001  0.001  1.000000
9   0.001000  0.001000  0.001  0.001  0.001  0.686062  0.001  0.001  0.897596
10  0.001000  0.001000  0.001  0.001  0.001  0.794686  0.001  0.001  0.001000

          9         10
0   0.001000  0.001000
1   0.001000  0.001000
2   0.001000  0.001000
3   0.001000  0.001000
4   0.001000  0.001000
5   0.686062  0.794686
6   0.001000  0.001000
7   0.001000  0.001000
8   0.897596  0.001000
9   1.000000  0.012274
```

```
10  0.012274  1.000000
```

Here, we can see that for predictors:

> II beta-HCG(mIU/mL)
> Follicle No. (R)
> Endometrium (mm)

the means are same.

For, columns:

> Age (yrs)
> BMI
> Waist:Hip Ratio
> Cycle(R/I)
> I beta-HCG(mIU/mL)
> BP _Diastolic (mmHg)
> Hb(g/dl)
> Follicle No. (L)

the means are different.

4.3 The Analysis of Categorical Data

To analyse this data, we perform the chi-square test which tells us the dependency of the variables; in this case, the response and the predictor variables.

To perform the test and check for the dependency of the PCOS affecting the regularity of the cycle, we apply the chi-square test.

Let the hypothesis testing be:

H_0 : PCOS affects the irregularity of cycle based on the number of days.

vs

H_1 : PCOS doesn't affect the irregularity of cycle based on the number of days.

In [58]:
```
crosstab, results, expected = rp.crosstab(data["PCOS (Y/N)"], data["Cycle(R/I)"],
                                          test= "chi-square",
                                          expected_freqs= True,
                                          prop= "cell")

crosstab
```

Out[58]:

	Cycle(R/I)			
Cycle(R/I)	**2**	**4**	**5**	**All**
PCOS (Y/N)				
0	57.06	10.41	0.00	67.47
1	15.06	17.29	0.19	32.53
All	72.12	27.70	0.19	100.00

Here, we get the chi-square table giving us an overview the calculative part.

In [59]:
```
results
```

Out[59]:

	Chi-square test	results
0	Pearson Chi-square (2.0) =	86.7215
1	p-value =	0.0000
2	Cramer's V =	0.4015

From, the above results we obtain the results.

From the results, we can see that the p-value is so small, that it just shows here as 0.

Thus, we know we need to accept H_0. But inorder to be very sure, we need to need the Cramer V results.

In [60]:
```
Cramers_v = results["results"][2]

if Cramers_v > 0.25:
    print("Very strong interpretation\nAccept H0")
elif Cramers_v > 0.15:
    print("Very strong interpretation\nnAccept H0")
elif Cramers_v > 0.10:
    print("Moderate interpretation\nReject H0")
elif Cramers_v > 0.05:
    print("Weak interpretation\nReject H0")
else:
    print("No or very weak interpretation\nReject H0")
```

```
Very strong interpretation
Accept H0
```

Since, the Cramer V we obtained is 0.4015, it is significantly higher than 0.25 which is a threshold to state that the interpretation that 'PCOS affects the irregularity of cycle based on the number of days' is very strong and thus is it definitely true.

Thus, we conclude that PCOS affects the irregularity of cycle based on the number of days.

4.4 Logistic Regression

After analysing the data well, I found out that my response variable is giving a binary response i.e. it is Yes or No. Hence, in order to perform regression, I performed Logistic Regression with 'PCOS' as my response variable and rest columns as my predictors.

In [132…

```
df_log = data

X = df_log.drop('PCOS (Y/N)',axis = 1)
X = pd.DataFrame(df_log)
Y = df_log['PCOS (Y/N)']

X_train, X_test, Y_train, Y_test = train_test_split(X,
                                                    Y,
                                                    random_state=0,
                                                    test_size= 0.2)

print("Count for each is --- ")
print("X_train: \t",len(X_train))
print("X_test : \t",len(X_test))
print("y_train: \t",len(Y_train))
print("y_test : \t",len(Y_test))
```

```
Count for each is ---
X_train:         430
X_test :         108
y_train:         430
y_test :         108
```

In [62]:

```
# Logistic Regression

clf = LogisticRegression(solver='liblinear',
                         random_state=0)
clf.fit(X_train, Y_train)

Y_predict = clf.predict(X_test)

from sklearn.metrics import accuracy_score, classification_report
score_test_ = accuracy_score(Y_test,Y_predict)
print(classification_report(Y_test,Y_predict))

print("Accuracy of Logistic Regression :", score_test_*100,"%")
```

```
              precision    recall  f1-score   support

           0       0.96      1.00      0.98        65
           1       1.00      0.93      0.96        43

    accuracy                           0.97       108
   macro avg       0.98      0.97      0.97       108
weighted avg       0.97      0.97      0.97       108

Accuracy of Logistic Regression : 97.22222222222221 %
```

Ridge Regression

In [63]:

```
# Ridge Regression

from sklearn.linear_model import Lasso
from sklearn.metrics import mean_squared_error as mse
from sklearn.metrics import mean_absolute_error as mae

df = data

X = df.drop(['PCOS (Y/N)'],axis=1)
y = df['PCOS (Y/N)']
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size=0.2,
                                                    random_state=0)

ridReg = Ridge()
ridReg.fit(X_train, y_train)

y_pred = ridReg.predict(X_test)
print("Accuracy of Ridge  Regression: ", mae(y_test, y_pred)*100,"%")
```

```
Accuracy of Ridge  Regression:  55.99695513265522 %
```

Lasso Regression

In [64]:
```python
# Lasso Regression
from sklearn.linear_model import Lasso
from sklearn.metrics import mean_squared_error as mse
from sklearn.metrics import mean_absolute_error as mae

df = data

X = df.drop(['PCOS (Y/N)'],axis=1)
y = df['PCOS (Y/N)']
X_train, X_test, y_train, y_test = train_test_split(X,
                                                     y,
                                                     test_size=0.2,
                                                     random_state=0)

lasReg = Lasso()
lasReg.fit(X_train, y_train)

y_pred = lasReg.predict(X_test)
print("Accuracy of Lasso Regression: ", mae(y_test, y_pred)*100,"%")
```

```
Accuracy of Lasso Regression:  41.35547989367577 %
```

- Accuracy of Logistic Regression: 97.22222222222221 %
- Accuracy of Ridge Regression : 55.99695513265522 %
- Accuracy of Lasso Regression : 41.35547989367577 %

Thus, the best performing model here is Logistic Regression.

In [71]:
```python
from sklearn.model_selection import cross_validate

lg_resultant = cross_validate(LogisticRegression(n_jobs=-1),
                              x_train,
                              y_train,
                              cv=5,
                              return_train_score=True)

print("Mean Training Score = ", lg_resultant['train_score'].mean()*100)
print("Mean Testing Score = ", lg_resultant['test_score'].mean()*100)
```

```
Mean Training Score =  86.16279069767442
Mean Testing Score =  85.81395348837208
```

In [134…
```python
# Fitting the model on test dataset

lr.fit(X_train,y_train)
y_pred = lr.predict(X_test)
mse(y_test,y_pred)

print("Accuracy = ",(1-mse(y_test,y_pred))*100,"%")
```

```
Accuracy =  81.4814814814815 %
```

Using the above features, logistic regression give an accuracy of 81.48% which is strange but considering the features that are removed makes some sense.

4.5 Resampling Methods

In resampling methods, I have used bootstrap and k-fold cross validation. These methods refit a model of interest to samples formed from the training set, in order to obtain additional information about the fitted model.

Bootstraping

In [81]:
```python
from scipy.stats import bootstrap

df = (data,)

#calculating 95% bootstrapped confidence interval for median
bootstrap_ci = bootstrap(df, np.median, confidence_level=0.95,
                         random_state=1, method='percentile')

#95% boostrapped confidence interval
print(bootstrap_ci.confidence_interval)
```

```
ConfidenceInterval(low=array([ 0.   , 30.   , 23.8  , 11.   ,  2.   ,  5.   ,  6.   , 10.   ,
        1.99 ,  0.89 ,  3.455, 25.3  , 80.   ,  5.   ,  5.   ,  8.2  ]), high=array([ 0.  , 32.  , 24.6 , 11.  ,  2.  ,  5.  ,  7.
, 60.8 ,  1.99,
        0.9 ,  4.13, 27.35, 80.  ,  6.  ,  6.5 ,  8.65]))
```

In [114…
```python
train_score = []
test_score = []
n_times = 10

df_boot = data
```

```
## Performing bootstrapping
for i in range(n_times):
    X = df_boot.drop('PCOS (Y/N)',axis=1)
    y = df_boot['PCOS (Y/N)']

    x_train, x_test, y_train, y_test = train_test_split(X,y, test_size=0.2, random_state=42+i)

    #choose from different tunable hyper parameters
    clf = Lasso().fit(x_train, y_train)
    y_pred = clf.predict(x_test)

    # Storing accuracy values
    train_score.append(mae(y_train, clf.predict(x_train)))
    test_score.append(mae(y_test, clf.predict(x_test)))

#################################################
# Result of all bootstrapping trials
print("Mean Training MSE = ", np.mean(train_score)*100)
print("Mean Testing MAE = ", np.mean(test_score)*100)

print("Accuracy = ",np.mean(test_score)*100,"%")
```

```
Mean Training MSE =  38.851662385248055
Mean Testing MAE =  39.71085863511652
Accuracy =  39.71085863511652 %
```

K-fold Cross Validation

In [83]:

```
# k-fold cross validation

import pandas as pd
from sklearn.model_selection import KFold
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

#Loading the dataset
X = df_normalize.drop(['PCOS (Y/N)'],axis=1)
y = df_normalize['PCOS (Y/N)']
#X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_state=0)

#Implementing cross validation

k = 10
kf = KFold(n_splits = k)
model = LogisticRegression(solver= 'liblinear')

acc_score = []

for train_index , test_index in kf.split(X):

    X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_state=0)

    model.fit(X_train,y_train)
    pred_values = model.predict(X_test)

    acc = accuracy_score(pred_values , y_test)
    acc_score.append(acc)

avg_acc_score = sum(acc_score)/k

print('Average accuracy : {} %'.format(avg_acc_score*100))
```

```
Average accuracy : 81.48148148148147 %
```

- Accuracy using bootstrap method: 39.71085863511652 %
- Accuracy using k-fold cross validation: 81.48148148148147 %

Thus, we can state that resampling is affecting the performance of the model.

4.6 Linear Model Selection and Regularization

Feature Selection

Here, I perform feature selection to get an overall idea of best features using forward pass.

In [65]:

```
# Feature selection

import statsmodels.api as sm
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error as mse
from sklearn.metrics import mean_absolute_error as mae
from sklearn.feature_selection import SelectKBest,f_classif
from sklearn.model_selection import train_test_split

aic_score = []
mae_score = []
mse_score = []
mse_arr = []
```

```python
df_sel = data

X_fs = df_sel.drop('PCOS (Y/N)',axis = 1)
X_fs = pd.DataFrame(df_sel)

Y_fs = df_sel['PCOS (Y/N)']

for i in range(1,15):

    select_clf = SelectKBest(score_func = f_classif,
                             k=i)
    select_clf.fit(X_fs, Y_fs)
    new_X = X_fs.iloc[:,select_clf.get_support()]

    x_train, x_test, y_train, y_test = train_test_split(new_X,
                                                        df_sel['PCOS (Y/N)'],
                                                        test_size=0.2,
                                                        random_state=1)

    clf_fs = LogisticRegression().fit(x_train, y_train)
    y_pred = clf_fs.predict(x_test)

    log_reg = sm.OLS(y_train.values.ravel(), x_train).fit()

    print("Number of Paramters : {}".format(i))
    print("Columns :", X.columns[i])
    print("AIC Score: {}".format(log_reg.aic))
    print("MSE =", mse(y_pred, y_test))
    print("MAE =", mae(y_pred, y_test))
    aic_score.append(log_reg.aic)
    mse_score.append(mse(y_pred, y_test))
    mae_score.append(mae(y_pred, y_test))

    mse_arr.append(mse(y_pred, y_test))

    print("-----------------------------------------------------------")
    print("")

print("The best column in dataset is ",X.columns[mse_arr.index(min(mse_arr))])
```

```
Number of Paramters : 1
Columns : BMI
AIC Score: -30263.96333120281
MSE = 3.5123255726528703e-32
MAE = 1.8555116291188959e-16
-----------------------------------------------------------

Number of Paramters : 2
Columns : Hb(g/dl)
AIC Score: -30704.20984908393
MSE = 5.780540488463939e-32
MAE = 1.6404045888692697e-16
-----------------------------------------------------------

Number of Paramters : 3
Columns : Cycle(R/I)
AIC Score: -29883.23663711966
MSE = 1.2302965351176464e-31
MAE = 3.1793298576503053e-16
-----------------------------------------------------------

Number of Paramters : 4
Columns : Cycle length(days)
AIC Score: -30173.44427440499
MSE = 1.3035071670685293e-31
MAE = 3.056211650858571e-16
-----------------------------------------------------------

Number of Paramters : 5
Columns : Marraige Status (Yrs)
AIC Score: -29551.18877287063
MSE = 2.204911891855477e-31
MAE = 4.0635469776378875e-16
-----------------------------------------------------------

Number of Paramters : 6
Columns : I beta-HCG(mIU/mL)
AIC Score: -29903.51931254325
MSE = 7.759640249783278e-32
MAE = 2.3020192707927777e-16
-----------------------------------------------------------

Number of Paramters : 7
Columns : II beta-HCG(mIU/mL)
AIC Score: -29419.93909404848
MSE = 2.6038979267929905e-31
MAE = 4.3052087954704492e-16
-----------------------------------------------------------

Number of Paramters : 8
Columns : Waist:Hip Ratio
AIC Score: -29973.79916428095
MSE = 2.5115925423417615e-31
MAE = 4.17264250889331e-16
-----------------------------------------------------------

Number of Paramters : 9
```

```
Columns : AMH(ng/mL)
AIC Score: -29066.733870550757
MSE = 2.1432411585291244e-31
MAE = 3.780980677944242e-16
-----------------------------------------------------------

Number of Paramters : 10
Columns : Vit D3 (ng/mL)
AIC Score: -29140.424449834896
MSE = 3.236540109986048e-32
MAE = 1.2710249678043814e-16
-----------------------------------------------------------

Number of Paramters : 11
Columns : BP _Diastolic (mmHg)
AIC Score: -28328.472544984914
MSE = 2.9032798263081315e-32
MAE = 1.3129061010978417e-16
-----------------------------------------------------------

Number of Paramters : 12
Columns : Follicle No. (L)
AIC Score: -27860.72510801184
MSE = 2.805233791413352e-31
MAE = 4.205227011814114e-16
-----------------------------------------------------------

Number of Paramters : 13
Columns : Follicle No. (R)
AIC Score: -28231.06652401563
MSE = 8.213204015290024e-32
MAE = 2.2034424499569708e-16
-----------------------------------------------------------

Number of Paramters : 14
Columns : Endometrium (mm)
AIC Score: -26197.585991861302
MSE = 3.6368557727753869e-27
MAE = 1.5458726125343902e-14
-----------------------------------------------------------

The best column in dataset is  Vit D3 (ng/mL)
```

Minimum mse is for col Vit D3 (ng/mL), hence it is the best choice.

In [66]:
```
select_clf = SelectKBest(score_func = f_classif,
                         k=10)
select_clf.fit(x_train, y_train)
```

Out[66]:
```
SelectKBest()
```

In [67]:
```
X = X.drop("Cycle length(days)",axis = 1)
print(X.columns)
print(X.columns.shape)
```

```
Index([' Age (yrs)', 'BMI', 'Hb(g/dl)', 'Cycle(R/I)', 'Marraige Status (Yrs)',
       'I beta-HCG(mIU/mL)', 'II beta-HCG(mIU/mL)', 'Waist:Hip Ratio',
       'AMH(ng/mL)', 'Vit D3 (ng/mL)', 'BP _Diastolic (mmHg)',
       'Follicle No. (L)', 'Follicle No. (R)', 'Endometrium (mm)'],
      dtype='object')
(14,)
```

In [68]:
```
select_clf.get_support().shape
```

Out[68]:
```
(14,)
```

The best features obtained are as below:

In [69]:
```
kept_features = pd.DataFrame({'Columns': X.columns,
                              'Kept': select_clf.get_support()})
kept_features
```

Out[69]:

	Columns	Kept
0	Age (yrs)	True
1	BMI	True
2	Hb(g/dl)	True
3	Cycle(R/I)	False
4	Marraige Status (Yrs)	True
5	I beta-HCG(mIU/mL)	True
6	II beta-HCG(mIU/mL)	True
7	Waist:Hip Ratio	False
8	AMH(ng/mL)	True
9	Vit D3 (ng/mL)	True
10	BP _Diastolic (mmHg)	False

	Columns	Kept
11	Follicle No. (L)	True
12	Follicle No. (R)	True
13	Endometrium (mm)	False

In [70]:

```python
y = df_sel['PCOS (Y/N)']

x_train, x_test, y_train, y_test = train_test_split(X,y, test_size=0.2, random_state=1)

lin_reg_kbest = sm.OLS(y_train.values.ravel(), x_train).fit()

lin_reg_kbest.summary()
```

Out[70]:

OLS Regression Results

Dep. Variable:	y	**R-squared (uncentered):**	0.668
Model:	OLS	**Adj. R-squared (uncentered):**	0.657
Method:	Least Squares	**F-statistic:**	59.86
Date:	Fri, 16 Dec 2022	**Prob (F-statistic):**	2.38e-90
Time:	17:26:08	**Log-Likelihood:**	-128.56
No. Observations:	430	**AIC:**	285.1
Df Residuals:	416	**BIC:**	342.0
Df Model:	14		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Age (yrs)	-0.0027	0.004	-0.694	0.488	-0.010	0.005
BMI	0.0092	0.004	2.299	0.022	0.001	0.017
Hb(g/dl)	0.0161	0.017	0.971	0.332	-0.017	0.049
Cycle(R/I)	0.1082	0.019	5.584	0.000	0.070	0.146
Marraige Status (Yrs)	-0.0017	0.004	-0.376	0.707	-0.010	0.007
I beta-HCG(mIU/mL)	-7.717e-06	6.89e-06	-1.119	0.264	-2.13e-05	5.83e-06
II beta-HCG(mIU/mL)	5.436e-06	1.18e-05	0.461	0.645	-1.77e-05	2.86e-05
Waist:Hip Ratio	-0.4913	0.267	-1.840	0.067	-1.016	0.034
AMH(ng/mL)	0.0067	0.003	2.366	0.018	0.001	0.012
Vit D3 (ng/mL)	1.429e-05	4.19e-05	0.341	0.733	-6.8e-05	9.66e-05
BP _Diastolic (mmHg)	-0.0039	0.003	-1.500	0.134	-0.009	0.001
Follicle No. (L)	0.0216	0.007	3.265	0.001	0.009	0.035
Follicle No. (R)	0.0429	0.006	6.904	0.000	0.031	0.055
Endometrium (mm)	0.0023	0.007	0.308	0.758	-0.012	0.017

Omnibus:	14.797	**Durbin-Watson:**	2.146
Prob(Omnibus):	0.001	**Jarque-Bera (JB):**	15.469
Skew:	0.461	**Prob(JB):**	0.000437
Kurtosis:	3.123	**Cond. No.**	5.69e+04

Notes:
[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[3] The condition number is large, 5.69e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

Using these features as components, we perform the **Principle Component Analysis** on total *11 components*.

In [122…

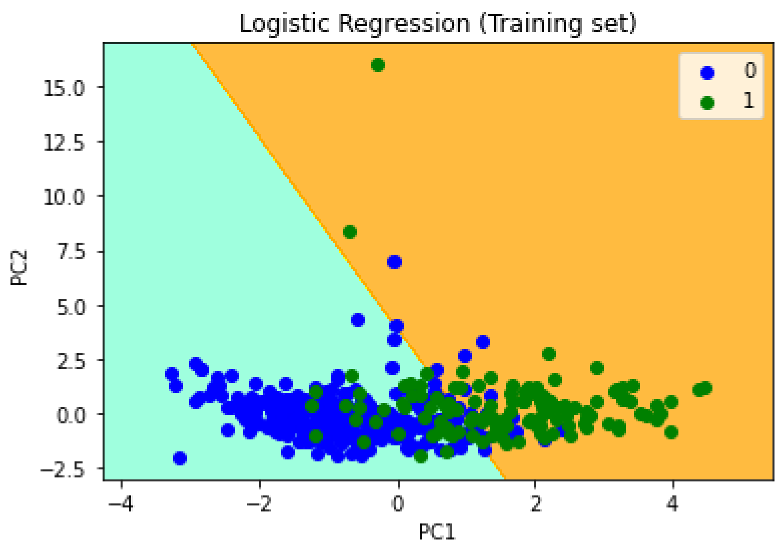
```python
# PCA

X = df_normalize.drop(['PCOS (Y/N)'],axis=1)
y = df_normalize['PCOS (Y/N)']
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_state=0)

# Splitting the X and Y into the Training set and Testing set
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

```python
# performing preprocessing part
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()

X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Applying PCA function on training
# and testing set of X component
from sklearn.decomposition import PCA

pca = PCA(n_components = 2)

X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)

explained_variance = pca.explained_variance_ratio_


from sklearn.linear_model import LogisticRegression

classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)

# Predicting the test set result using
# predict function under LogisticRegression
y_pred = classifier.predict(X_test)

# making confusion matrix between
# test set of Y and predicted value.
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)

TP = cm[0][0]
FN = cm[0][1]
FP = cm[1][0]
TN = cm[1][1]

err = (FP+FN)/(TP+TN+FP+FN)

print("Accuracy =",(1-err)*100,"%")

# Predicting the training set
# result through scatter plot
from matplotlib.colors import ListedColormap

X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1,
                                        stop = X_set[:, 0].max() + 1, step = 0.01),
                                        np.arange(start = X_set[:, 1].min() - 1,
                                        stop = X_set[:, 1].max() + 1, step = 0.01))

py.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
                        X2.ravel()]).T).reshape(X1.shape), alpha = 0.75,
                        cmap = ListedColormap(('aquamarine', 'yellow', 'orange')))

py.xlim(X1.min(), X1.max())
py.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y_set)):
        py.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                                c = ListedColormap(('blue', 'green', 'red'))(i), label = j)

py.title('Logistic Regression (Training set)')
py.xlabel('PC1') # for Xlabel
py.ylabel('PC2') # for Ylabel
py.legend() # to show legend

# show scatter plot
py.show()

# Visualising the Test set results through scatter plot
from matplotlib.colors import ListedColormap

X_set, y_set = X_test, y_test

X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1,
                                        stop = X_set[:, 0].max() + 1, step = 0.01),
                                        np.arange(start = X_set[:, 1].min() - 1,
                                        stop = X_set[:, 1].max() + 1, step = 0.01))

py.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
                        X2.ravel()]).T).reshape(X1.shape), alpha = 0.75,
                        cmap = ListedColormap(('aquamarine', 'yellow', 'orange')))

py.xlim(X1.min(), X1.max())
py.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y_set)):
        py.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                                c = ListedColormap(('blue', 'green', 'red'))(i), label = j)
```

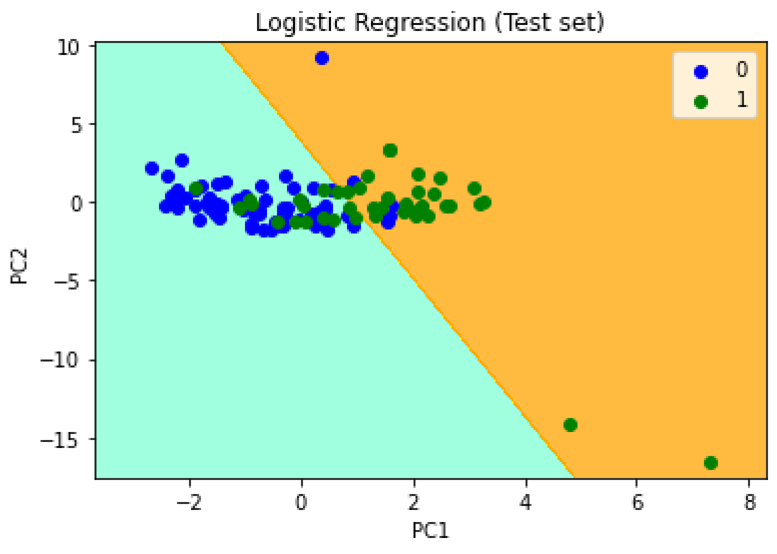
```
# title for scatter plot
py.title('Logistic Regression (Test set)')
py.xlabel('PC1') # for Xlabel
py.ylabel('PC2') # for Ylabel
py.legend()

# show scatter plot
```

```
Accuracy = 81.4814814814815 %
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in ca
se its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array with a single row if you int
end to specify the same RGB or RGBA value for all points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in ca
se its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array with a single row if you int
end to specify the same RGB or RGBA value for all points.
```



```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in ca
se its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array with a single row if you int
end to specify the same RGB or RGBA value for all points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in ca
se its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array with a single row if you int
end to specify the same RGB or RGBA value for all points.
```



Accuracy obtained using Principle Component Analysis is 81.4814814814815 %

4.7 Moving Beyond Linearity

We applied above models assuming linearity but in reality it is not linear. Hence, we apply polynomial regression.

Polynomial Regression

In [135…

```
# Moving Beyond Linearity

from sklearn.preprocessing import PolynomialFeatures

df_mbl = data

for i in range(2,6):
    X = df_mbl.drop(['PCOS (Y/N)'],axis=1)
    y = df_mbl['PCOS (Y/N)']
    X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_state=0)

    poly = PolynomialFeatures(degree = i)
    X_train = poly.fit_transform(X_train)
    X_test = poly.fit_transform(X_test)


    poly.fit(X_train, y_train)
    linReg = LogisticRegression()
    linReg.fit(X_train, y_train)

    y_pred = linReg.predict(X_test)
    print("for degree : {}".format(i))
    print("Accuracy : ", (1-mae(y_test, y_pred))*100,"%")
    print("===========================")
```

```
for degree : 2
Accuracy :  63.58024691358024 %
============================
for degree : 3
Accuracy :  65.4320987654321 %
============================
for degree : 4
Accuracy :  62.96296296296296 %
============================
for degree : 5
Accuracy :  62.96296296296296 %
============================
```

Here, we can notice that most accuracy of 65.4320987654321 % can be obtained for degree-3 polynomial which is worse than rest models obtained.

Generalised Additive Model

Since, my response variable in data is binary, I applied Logistic generalised additive model to get the best performance.

In [86]:

```
import pandas as pd
from pygam import LogisticGAM

df = data

# using the best features
X = df[[' Age (yrs)',
        'BMI',
        'Hb(g/dl)',
        'Cycle length(days)',
        'AMH(ng/mL)',
        'Marraige Status (Yrs)',
        'I beta-HCG(mIU/mL)',
        'Follicle No. (L)',
        'Follicle No. (R)',
        'Vit D3 (ng/mL)']];

y = df["PCOS (Y/N)"];
gam = LogisticGAM().fit(X, y);
```

```
did not converge
```

In [87]:

```
gam.summary()
```

```
LogisticGAM
=============================================== ==========================================================
Distribution:                       BinomialDist Effective DoF:                                     52.3743
Link Function:                         LogitLink Log Likelihood:                                  -143.0663
Number of Samples:                           538 AIC:                                              390.8811
                                                 AICc:                                             402.8829
                                                 UBRE:                                               2.8044
                                                 Scale:                                                 1.0
                                                 Pseudo R-Squared:                                   0.5784
==========================================================================================================
Feature Function                  Lambda               Rank         EDoF         P > x        Sig. Code
================================= ==================== ============ ============ ============ ============
s(0)                              [0.6]                20           10.5         4.26e-01
s(1)                              [0.6]                20           8.1          2.64e-01
s(2)                              [0.6]                20           6.3          0.00e+00     ***
s(3)                              [0.6]                20           6.3          6.41e-03     **
s(4)                              [0.6]                20           4.0          3.79e-02     *
s(5)                              [0.6]                20           4.5          9.84e-01
s(6)                              [0.6]                20           2.5          9.97e-01
s(7)                              [0.6]                20           4.3          2.10e-01
s(8)                              [0.6]                20           4.8          9.62e-06     ***
s(9)                              [0.6]                20           1.1          8.73e-01
intercept                                              1            0.0          9.60e-01
==========================================================================================================
Significance codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

WARNING: Fitting splines and a linear function to a feature introduces a model identifiability problem
         which can cause p-values to appear significant when they are not.

WARNING: p-values calculated in this manner behave correctly for un-penalized models or models with
         known smoothing parameters, but when smoothing parameters have been estimated, the p-values
         are typically lower than they should be, meaning that the tests reject the null too readily.
```

In [88]:

```
print("Accuracy =",gam.accuracy(X, y)*100,"%")
```

```
Accuracy = 89.77695167286245 %
```

Here, we observe that the Logistic GAM model gives an accuracy of almost 90% but it is still worse than Logistic Regression.

Conclusion

Thus, we found out that the PCOS dataset is a not normally distributed and hence we apply the non-normal and non-parametric tests to get better analysis. We found out that the dataset is different. The irregularity of menstrual cycles depending on days is affect partially by PCOS.

The best model that fits this dataset is Logistic Regression. Since, using attributes obtained in feature selection does not help completely, I conclude that all attributes from the dataset contribute equally in getting valuable insight into the analysis.

References

- SHREYAS VEDPATHAK,"Dataset",https://www.kaggle.com/datasets/shreyasvedpathak/pcos-dataset, Kaggle.
- İLAYDA DURATNIR,"PCOS Classification", https://www.kaggle.com/code/ilaydadu/pcos-classification.
- KARNIKA KAPOOR, "PCOS Diagnosis", https://www.kaggle.com/code/karnikakapoor/pcos-diagnosis.