SHIV NADAR UNIVERSITY
DELHI NCR

EED363 Applied Machine Learning

End Term Project Report

Project Group No. : 11

Group Members:

1. **Ajay Gokul S - 1810110014**
2. **Debanjali Saha - 1810110059**

Dataset : Open-source dataset available at Kaggle Repository on PCOS
https://www.kaggle.com/prasoonkottarathil/polycystic-ovary-syndrome-pcos

Our code : Publicly accessible GitHub Repository called PCOS_Project
https://github.com/debsa2000/PCOS_Project

Table of Contents

1. Abstract 4

2. Introduction 5
   - 2.1 Problem statement 5
   - 2.2 Domain background 5
   - 2.3 Motivation 6
   - 2.4 Data set 7
   - 2.5 Literature review 7

3. Project Overview 8
   - 3.1 Working Methodology Flow Chart 8
   - 3.2 Data Exploration 9
     - 3.2.1 Explanation of Data 9
     - 3.2.2 Statistical Properties 10
   - 3.3 Data Preprocessing and Transformation 11
     - 3.3.1 Data cleaning 11
     - 3.3.2 Data visualization 15
     - 3.3.3 Outlier analysis 18
   - 3.4 Feature Selection 22
     - 3.4.1 Univariate selection 22
     - 3.4.2 Feature importance 24
     - 3.4.3 Correlation matrix with heatmap 26
   - 3.5 Feature extraction 29
     - 3.5.1 Principal Component Analysis 29

4. Training the Model 31
   - 4.1 Classification & splitting of data 31
   - 4.2 Confusion matrix 32
   - 4.3 ROC curve 33
   - 4.4 Performance metrics 33

5. Classifiers 35
   - 5.1 Naive Bayes Classifier 35
     - 5.1.1 Model Building 35
     - 5.1.2 Performance Measures (with all features) 36
     - 5.1.3 Performance Measures after feature selection 38
   - 5.2 k-Nearest Neighbours Classifier 40
     - 5.2.1 Model Building 40
     - 5.2.2 Performance Measures (with all features) 41
     - 5.2.3 Performance Measures after feature selection 44
   - 5.3 Decision Tree Classifier 46
     - 5.3.1 Model Building 46
     - 5.3.2 Performance Measures (with all features) 46

5.3.3 Performance Measures after feature selection 49
5.4 Ensemble Classifier - Random Forest (Majority vote) 51
5.4.1 Model Building 51
5.4.2 Performance Measures (with all features) 52
5.4.3 Performance Measures after feature selection 54
5.5 Ensemble Classifier - Adaboost (Boosting Algorithm) 56
5.3.1 Model Building 56
5.5.2 Performance Measures (with all features) 57
5.5.3 Performance Measures after feature selection 60
5.6 Logistic Regression 62
5.6.1 Justification for not using Linear Regression 62
5.6.2 Model Building 63
5.6.3 Performance Measures (with all features) 64
5.6.4 Performance Measures after feature selection 67
5.6.5 Log-Loss of Binary Cross Entropy 69
5.7 Support Vector Machine (SVM) 72
5.7.1 Model Building 73
5.7.2 Performance Measures (with all features) 74
5.7.3 Performance Measures after feature selection 76
5.8 Experimenting with Neural Network 79
5.8.1 Multi Layer Perceptron Transfer 80
5.8.2 Performance Measures 81

6. Clustering 83
6.1 Overview of our way 83
6.2 KMeans Clustering 83
6.3 Implementation 84

7. Results and Conclusions 87
7.1 Tabulated Results 87
7.2 Inference 87
7.3 Conclusion 89
7.4 Comparison 90

8. Acknowledgement 91

9. References 92

1. Abstract

This end-term project report is on the detection and prediction of an endocrine disorder affecting women of reproductive age, known as Polycystic Ovarian Syndrome (PCOS) or Polycystic Ovarian Disorder (PCOD). It begins by outlining the problem statement and a brief introduction to the domain providing the necessary understanding of PCOS and its relevance. After discussing the literature reviewed by us of the relevant machine learning research in this field, it moves to discuss the overall working methodology of this end-term project. We start off with the detailed 'data exploration' which includes explanation of the data and analysing its statistical properties. This is followed by some important 'data preprocessing' steps like data cleaning, handling missing data and outlier analysis. Following this is the 'data transformation' process where interpretation and visualization of data along with detection of important features, that is, feature selection and feature extraction is done.

These steps are in the lead up to the important task of training a machine learning model with an efficient data set to predict PCOS. In this project, classification models were built and trained with various algorithms like Naive Bayes, KNN, Logistic Regression, SVM and Ensemble. These models were first tested with all features and then tested after feature selection. The corresponding PCOS prediction accuracy and other performance measures are compared, analysed and summarised. In addition to this we have experimented with Neural Network and also illustrated the process of clustering. Observations, detailed note of our theoretical understanding, analysis of the obtained results and conclusions are made for all the graphical and numerical data obtained throughout this report.

Keywords - Data exploration, Data preprocessing, Data transformation, Classifier algorithms and PCOS

2. Introduction

2.1 Problem Statement

Detection and prediction of the endocrine disorder known as Polycystic Ovarian Syndrome (PCOS) affecting women of reproductive age is a relatively less explored field of research in Machine Learning. According to the New England Journal of Medicine, PCOS affects around 5-10% of women in reproductive age. (McCartney 2016). [1] Early detection of this disorder can ease the burden on the health care system in terms of cost and time that is spent in numerous analyses of clinical and metabolic parameters. The diagnosis of PCOS is made simple by the symptom-oriented and biological test-based application of standardized markers. Research is also ongoing in image-based prediction using radiological scan of ovaries. However, due to the scope of this course, we have decided to only use numeric data in our project to diagnose and predict PCOS.

2.2 Domain background

PCOD is the most common endocrine disorder among women during their childbearing years. The exact cause of PCOS is unknown in spite of it being a very common condition that affects how a woman's ovaries work. It is a hormonal disorder causing enlarged ovaries with small cysts on the outer edges.

However there are some features which are characteristic of this disorder

- Irregular menstrual cycles (irregular or skipped periods)
- High levels of male hormones (excess androgen)
- Polycystic ovaries(multiple cysts in the ovaries)

Etymology- The term 'polycystic ovaries' literally means ovaries containing "many cysts." or otherwise known as follicles, which are underdeveloped sacs in which eggs develop. The term 'cysts' again is not to be confused with true cysts. It does not lead to any form of ovarian cancer or get bigger, but instead they are follicles that have not matured to be ovulated. Therefore, when a woman has PCOS it means these sacs are unable to release eggs, therefore, affecting one's ovulation and menstrual cycles.

The lack of ovulation alters levels of hormones estrogen, progesterone, follicle stimulating hormone (FSH), and luteinizing hormone (LH). The ovaries also produce a small amount of male hormones called androgens. Estrogen and progesterone levels are lower than usual, while androgen levels are higher than usual. Extra male hormones disrupt the menstrual cycle, so women with PCOS get fewer periods than usual. Follicle-stimulating hormone (FSH) and luteinizing hormone (LH) control ovulation. FSH

stimulates the ovary to produce a follicle (a sac that contains an egg) and then LH triggers the ovary to release a mature egg.

Although there is no cure for PCOS, the symptoms can be treated. If left undiagnosed it can be a major issue for women of childbearing age leading to complications like miscarriage, infertility or gestational diabetes, This can also lead to hormonal imbalance, metabolic disorders, higher risk of developing type 2 diabetes and high cholestral levels in later amongst other things which can affect the overall health of the woman.

There are several signs and symptoms using which a diagnosis of PCOS can be made. This includes levels of certain male hormones, obesity, frequency of irregularity in menstrual cycle, acne and hirsutism. The heterogeneity of symptoms associated with PCOS makes the detection all the more laborious. Therefore, it necessitates formation of an optimal model which utilizes efficient and necessary clinical and metabolic parameters in making the right prediction.

Lifestyle interventions are the first treatments doctors recommend for PCOS, and they often work well. Weight loss can treat PCOS symptoms and improve the odds of getting pregnant. Diet and aerobic exercise are two effective ways to lose weight. Medicines are an option if lifestyle changes don't work. Birth control pills and metformin can both restore more normal menstrual cycles and relieve PCOS symptoms.

2.3 Motivation

The primary motivation for choosing the healthcare domain of diagnosis of Polycystic Ovarian Disorder as the course project theme was that one of the team members has been diagnosed with this disorder in her early years of teenage and this being a chronic and lifestyle disorder continues to be a major concern to be cured in her life. It needs awareness among everyone, especially among the women who are affected by it so that it could be treated by whatever lifestyle changes are feasible and hence needs to be predicted efficiently. Being so common among women today, it often either goes unnoticed if symptoms remain to be not so severe or is unknown to the patient herself. Especially because of this being related to female reproductive processes such as menstruation, it remains to be a taboo topic that needs further research and analysis. There hasn't been much technical work that has been done on this topic except the few literature that we have reviewed. It would indeed be an extremely useful work and novel area of research to apply Machine Learning on huge Gynecological datasets of women and predict the presence of PCOS in them in order to help cure the widespread disorder in women today.

2.4 Data set

To build this machine learning model we intend to apply several machine learning algorithms and techniques to analyse data and make predictions. The data set used for this is from an open source kaggle repository. It has physical and clinical parameters to determine PCOS obtained through a patient survey of 541 women during clinical consultations collected from 10 different hospitals across the state of Kerala, India. There are two parts to the dataset; The first one ('without infertility' data) has details of the samples for 43 attributes. The second part has data 'with infertility', which we plan to use after we have covered more material on Machine Learning in the lectures, essentially for the final report of this project.

2.5 Literature review

PCOS detection using machine learning is a relatively new area of research primarily due to the taboo associated with this topic for many years. However, there are some significant researchers whose contributions to this field have sparked interest. The "i-HOPE: Detection And Prediction System For Polycystic Ovary Syndrome (PCOS) Using Machine Learning Techniques" published in the Institute of Electrical and Electronics Engineers (IEEE) journal in 2019 by Denny et al. using the same data set as the one we are using for this project was the first major work of research in this field. They applied various machine learning techniques like Naives Bayes, KNN, CART, Random Forest, SVM and logistic regression and analysed the accuracy levels of these algorithms in the prediction of PCOS. [2]

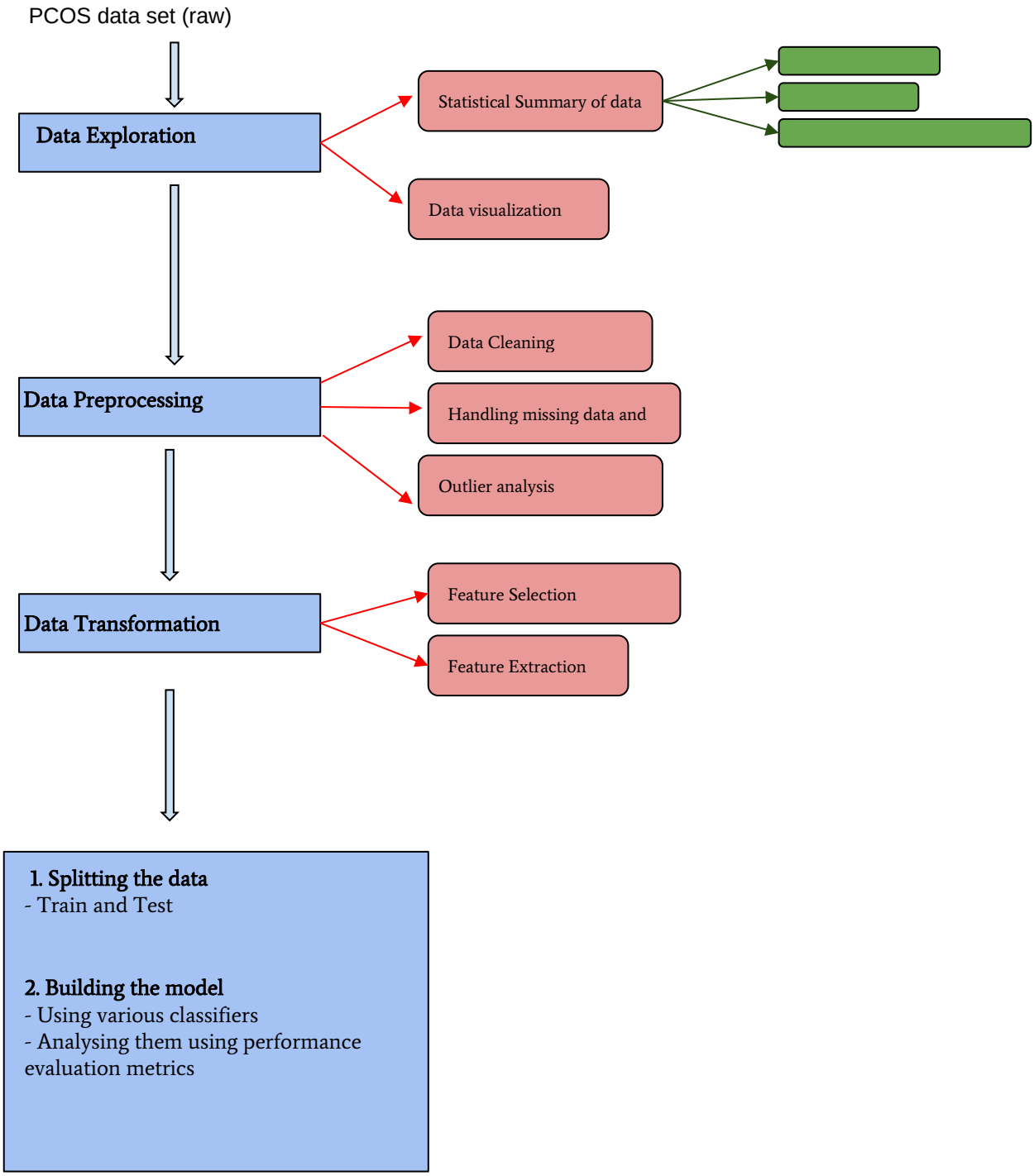
Following this, in the next year 2020, another paper was published in the IEEE journal by Bharti et al. applying an univariate feature selection algorithm to rank and identify which among the 45 attributes (including Sl no. and Patient File no.) is the most important feature in PCOS prediction. It also analysed the accuracy of classifiers such as gradient boosting and RFLR in predicting PCOS. [3]

Apart from these there are several other papers published in other journals which use the same data set to make some important conclusions and calculations. A paper by Munjal et al. in late 2020 published by the International Journal of Scientific Research calculated precision, confusion matrix and F1 score. [4] The works of Tanwani and Hassan et al in various international journals validated the accuracy of several algorithms. [5]

3. Project overview

3.1 Working Methodology Flow Chart

We have represented the methodology used in this end-term project report with the help of the following simple block diagram.



3.2 Data Exploration

3.2.1 Explanation of data

The various features of this data set is explained along with the type of data it contains. The details of the range of attributes are discussed after data cleaning process in section

Feature	Explanation	Type of Data
Sl. No.	Serial number of data set	Integer
Patient File No.	Case no for patient privacy	Integer
PCOS	Patient has PCOS (Yes=1/No=0)	Integer
Age (yrs)	How old the patient is (yrs)	Integer
Weight (Kg)	How much does the patient weigh? (Kg)	Integer
Height(Cm)	How tall is the patient? (cm)	Integer
BMI	Body mass index (BMI) is a measure of body fat based on height and weight	Float
Blood Group	A+ = 11 A- = 12 B+ = 13 B- = 14 O+ =15 O- = 16 AB+ =17 AB- = 18	Integer
Pulse rate	Number of times one's heart beats in 1 minute, also called heart rate (bpm)	Integer
RR	Respiratory rate is the number of breaths a patient takes per minute (breaths/min)	Float
Hb	Hemoglobin count in blood (g/dl)	Float
Cycle(R/I)	The regularity of the menstrual cycle (Regular = 2/ Irregular = 4)	Integer
Cycle length(days)	The length of menstrual cycles (in days)	Integer
Marriage Status (Yrs)	How long has the patient been married? (in years)	Integer
Pregnant(Y/N)	Is the patient pregnant? (Yes =1 / No = 0)	Integer
No. of aborptions	No. of abortions	Integer
FSH	Follicle stimulating hormone level (mIU/mL)	Float
LH	Luteinizing hormone level (mIU/mL)	Float
FSH/LH	FSH/LH	Float
Hip	Size of hip (inch)	Integer
Waist	Size of waist (inch)	Integer
Waist:Hip Ratio	size of waist/size of hip	Float
TSH	Thyroid stimulating (mIU/L)	Float
AMH	Anti-Mullerian hormone level (ng/mL)	Float
PRL	Prolactin level (ng/mL)	Float
Vit D3	Level of Vitamin D3 (ng/mL)	Float
PRG	Serum progesterone level (ng/mL)	Float

RBS	Random Glucose testing for blood sugar level (mg/dl)	Integer
Weight gain(Y/N)	Have you gained weight recently? (Yes =1 / No = 0)	Integer
hair growth(Y/N)	Is there hair growth? (Yes =1 / No = 0)	Integer
Skin darkening (Y/N)	Is the patient experiencing skin darkening ? (Yes =1 / No = 0)	Integer
Hair loss(Y/N)	Is there hair loss? (Yes =1 / No = 0)	Integer
Pimples(Y/N)	Presence of pimples? (Yes =1 / No = 0)	Integer
Fast food (Y/N)	Regular fast food habit? (Yes =1 / No = 0)	Integer
Reg.Exercise(Y/N)	Do you exercise regularly? (Yes =1 / No = 0)	Integer
BP _Systolic	Systolic blood pressure, measure of pressure in arteries when heart beats (mmHg)	Integer
BP _Diastolic	Diastolic blood pressure, measure of pressure in arteries when heart rests between beats (mmHg)	Integer
Follicle No. (L)	The number of follicles in Left ovary	Integer
Follicle No. (R)	The number of follicles in Right ovary	Integer
Avg. F size (L)	The average follicle size in Left ovary (in mm)	Integer
Avg. F size (R)	The average follicle size in Right ovary (in mm)	Integer
Endometrium	The thickness of the inner lining of the uterus (in mm)	Float

3.2.2 Statistical Properties

An important part of the data exploration exercise is to have an understanding of the minimum, maximum, mean and standard deviation of each feature.

```
27  # Generating a csv file that describes the Data
28  # (Count, Mean, S.D, Min and Max for each column)
29  df.describe().T.to_csv("my_description.csv")
```

There are two types of features - categorical and numeric. The following table highlights the only numeric features types.

Feature	Mean	Standard Deviation	Minimum	Maximum
Age (yrs)	31.43042672	5.414104704	20	48
Weight (Kg)	59.63729128	11.0418737	31	108
Height(Cm)	156.4717922	6.039703177	137	180
BMI	24.31540817	4.061713211	12.41788175	38.9
Blood Group	13.80148423	1.84318236	11	18
Pulse rate(bpm)	73.25231911	4.437857709	13	82
RR (breaths/min)	19.23747681	1.68725981	16	28
Hb(g/dl)	11.16155844	0.8680192498	8.5	14.8
Cycle(R/I)	2.554730983	0.8993341568	2	5

Cycle length(days)	4.93877551	1.491599521	0	12
Marriage Status (Yrs)	7.685899814	4.80690582	0	30
No. of aborptions	0.2894248609	0.6936377381	0	5
FSH(mIU/mL)	14.64723748	217.4238039	0.21	5052
LH(mIU/mL)	6.49251577	86.83341068	0.02	2018
FSH/LH	6.902837505	60.80437026	0.002145689	1372.826087
Hip(inch)	37.99257885	3.974820918	26	48
Waist(inch)	33.83858998	3.600123224	24	47
Waist:Hip Ratio	0.8918409162	0.04633136268	0.755555556	0.979166667
TSH (mIU/L)	2.98512616	3.761969744	0.04	65
PRL(ng/mL)	24.35207792	14.97987015	0.4	128.24
Vit D3 (ng/mL)	50.01556215	346.8455989	0	6014.66
PRG(ng/mL)	0.6120797774	3.815878914	0.047	85
RBS(mg/dl)	99.86493506	18.58750066	60	350
BP _Systolic (mmHg)	114.6419295	7.39107927	12	140
BP _Diastolic (mmHg)	76.93506494	5.574891813	8	100
Follicle No. (L)	6.116883117	4.223239908	0	22
Follicle No. (R)	6.643784787	4.444537744	0	20
Avg. F size (L) (mm)	15.01818182	3.565130023	0	24
Avg. F size (R) (mm)	15.44966605	3.319807442	0	24
Endometrium (mm)	8.472115028	2.166570873	0	18

3.3 Data Preprocessing and Transformation

3.3.1 Data cleaning

It is necessary to highlight certain changes that had to be made in the original open source data set from Kaggle

- Some entries had null entries, these have been taken care of in the Data cleaning section 3.2..2
- An unnamed column 42 with no entries and the 'Patient File no.' which is redundant for our purpose have been deleted.
- The original downloadable data set could not coherently calculate BMI due to the use of an inexistent function "DIVIDE", starting from the second row, therefore, the necessary change has been made to calculate it from weight and height attributes.

We follow a three step process to identify null entries and take appropriate action:

1. Creation of Data Frame - Initially we had 541 rows and 43 columns (as can be seen from output of following code snippet) in the data-frame that we created using pandas library in Python..

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

df = pd.read_csv(
    'C:/Users/Dayanand Saha/Documents/PCOS_Project/Data/data without infertility.csv')

print(df)
```

```
Python - ExploratoryDataAnalysis.py:5 ✔

     Sl. No  Patient File No.  ...  Endometrium (mm)  Unnamed: 42
0         1             10001  ...               8.5          NaN
1         2             10002  ...               3.7          NaN
2         3             10003  ...              10.0          NaN
3         4             10004  ...               7.5          NaN
4         5             10005  ...               7.0          NaN
..      ...               ...  ...               ...          ...
536     537             10537  ...               6.7          NaN
537     538             10538  ...               8.2          NaN
538     539             10539  ...               7.3          NaN
539     540             10540  ...              11.5          NaN
540     541             10541  ...               6.9          NaN

[541 rows x 43 columns]
[Finished in 3.67s]
```

2. Removal of Redundant columns - We deleted the two redundant columns using the code that follows:
   - ❏ Unnamed column 42 (last column)
   - ❏ 'Patient File no.'

```
# Deleting redundant columns
del df["Unnamed: 42"]
del df['Patient File No.']
print(df)
```

```
Python - ExploratoryDataAnalysis.py:8 ✔

     Sl. No  PCOS (Y/N)  ...  Avg. F size (R) (mm)  Endometrium (mm)
0         1           0  ...                  18.0               8.5
1         2           0  ...                  14.0               3.7
2         3           1  ...                  20.0              10.0
3         4           0  ...                  14.0               7.5
4         5           0  ...                  14.0               7.0
..      ...         ...  ...                   ...               ...
536     537           0  ...                  10.0               6.7
537     538           0  ...                  18.0               8.2
538     539           0  ...                   9.0               7.3
539     540           0  ...                  16.0              11.5
540     541           1  ...                  18.0               6.9

[541 rows x 41 columns]
[Finished in 3.537s]
```

3. Missing Value Detection - Next, we identified and printed the rows and columns which have entries with NULL values (NaN) in the entire dataset (as can be seen in the following code). It was found that the feature Fast Food(Y/N) has NaN value for the 158th sample and feature Marraige (sic) status has NaN value for the 460th sample. The respective rows are deleted instead of giving an input from our side. We are herewith left with 539 columns and 41 rows.

```
17  # Identifying entires with NULL (NaN)
18  null_columns = df.columns[df.isnull().any()]
19  df[null_columns].isnull().sum()
20  print(df[df.isnull().any(axis=1)][null_columns])
21
22  # Deleting rows with NULL entries
23  df = df.dropna()
24  df.drop(df.index[305])
25  print(df)
26
```

Python - ExploratoryDataAnalysis.py:21 ✔

```
     Marraige Status (Yrs)  Fast food (Y/N)
156                    5.0              NaN
458                    NaN              0.0
     Sl. No  PCOS (Y/N)  ...  Avg. F size (R) (mm)  Endometrium (mm)
0         1           0  ...                  18.0               8.5
1         2           0  ...                  14.0               3.7
2         3           1  ...                  20.0              10.0
3         4           0  ...                  14.0               7.5
4         5           0  ...                  14.0               7.0
..      ...         ...  ...                   ...               ...
536     537           0  ...                  10.0               6.7
537     538           0  ...                  18.0               8.2
538     539           0  ...                   9.0               7.3
539     540           0  ...                  16.0              11.5
540     541           1  ...                  18.0               6.9

[539 rows x 41 columns]
[Finished in 3.403s]
```

Having further look at the data to check any other discrepancies that may exist using the following code:

```
27  print(df.info())
```

```
Python - ExploratoryDataAnalysis.py:27 ✔

<class 'pandas.core.frame.DataFrame'>
Int64Index: 539 entries, 0 to 540
Data columns (total 41 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Sl. No                539 non-null    int64
 1   PCOS (Y/N)            539 non-null    int64
 2    Age (yrs)            539 non-null    int64
 3   Weight (Kg)           539 non-null    float64
 4   Height(Cm)            539 non-null    float64
 5   BMI                   539 non-null    float64
 6   Blood Group           539 non-null    int64
 7   Pulse rate(bpm)       539 non-null    int64
 8   RR (breaths/min)      539 non-null    int64
 9   Hb(g/dl)              539 non-null    float64
 10  Cycle(R/I)            539 non-null    int64
 11  Cycle length(days)    539 non-null    int64
 12  Marraige Status (Yrs) 539 non-null    float64
 13  Pregnant(Y/N)         539 non-null    int64
 14  No. of aborptions     539 non-null    int64
 15  FSH(mIU/mL)           539 non-null    float64
 16  LH(mIU/mL)            539 non-null    float64
 17  FSH/LH                539 non-null    float64
 18  Hip(inch)             539 non-null    int64
 19  Waist(inch)           539 non-null    int64
 20  Waist:Hip Ratio       539 non-null    float64
 21  TSH (mIU/L)           539 non-null    float64
 22  AMH(ng/mL)            539 non-null    object
 23  PRL(ng/mL)            539 non-null    float64
 24  Vit D3 (ng/mL)        539 non-null    float64
 25  PRG(ng/mL)            539 non-null    float64
 26  RBS(mg/dl)            539 non-null    float64
 27  Weight gain(Y/N)      539 non-null    int64
 28  hair growth(Y/N)      539 non-null    int64
 29  Skin darkening (Y/N)  539 non-null    int64
 30  Hair loss(Y/N)        539 non-null    int64
 31  Pimples(Y/N)          539 non-null    int64
 32  Fast food (Y/N)       539 non-null    float64
 33  Reg.Exercise(Y/N)     539 non-null    int64
 34  BP _Systolic (mmHg)   539 non-null    int64
 35  BP _Diastolic (mmHg)  539 non-null    int64
 36  Follicle No. (L)      539 non-null    int64
 37  Follicle No. (R)      539 non-null    int64
 38  Avg. F size (L) (mm)  539 non-null    float64
 39  Avg. F size (R) (mm)  539 non-null    float64
 40  Endometrium (mm)      539 non-null    float64
dtypes: float64(18), int64(22), object(1)
memory usage: 193.0+ KB
None
[Finished in 4.279s]
```

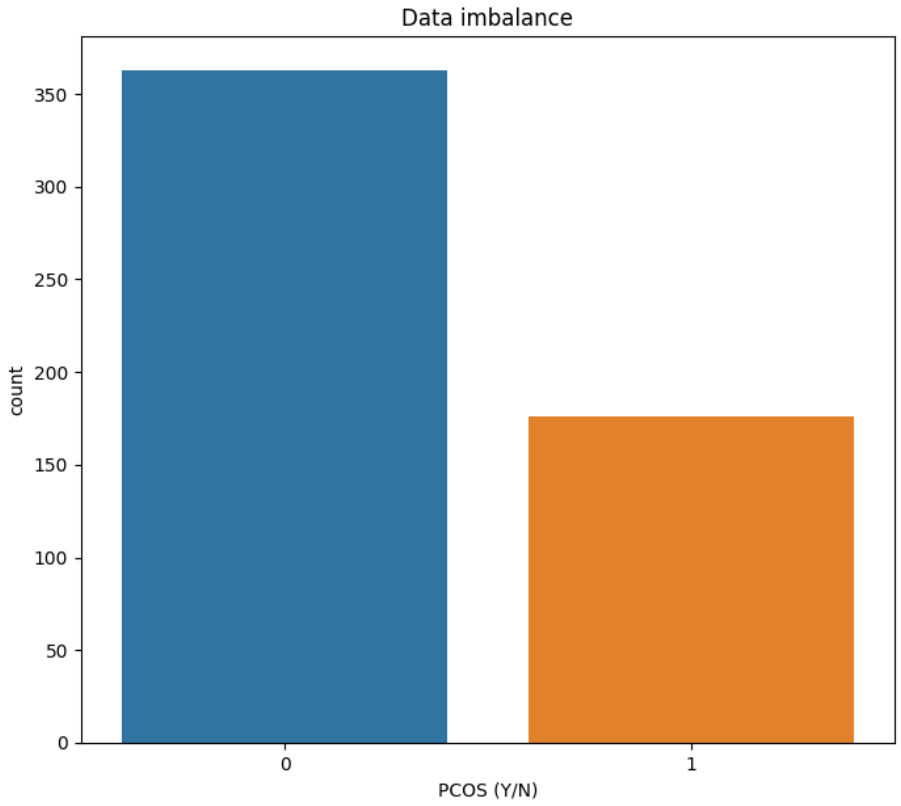
It was found that some numeric data is saved as strings. We check this in the following code first. In this database the data type “object” is a numeric value saved as string. As it can be seen the 22nd parameter ‘AMH (ng/mL)’ has an object type (in above terminal snapshot). Hence, we convert it to numeric value as follows and recheck its data type.

```
29  # Checking datatype
30  print(df["AMH(ng/mL)"].head())
31
32  # Converting to numeric value.
33  df["AMH(ng/mL)"] = pd.to_numeric(df["AMH(ng/mL)"], errors='coerce')
34
35  # Rechecking datatype
36  print(df["AMH(ng/mL)"].head())
```

```
Python - ExploratoryDataAnalysis.py:35 ✔

0    2.07
1    1.53
2    6.63
3    1.22
4    2.26
Name: AMH(ng/mL), dtype: object
0    2.07
1    1.53
2    6.63
3    1.22
4    2.26
Name: AMH(ng/mL), dtype: float64
[Finished in 4.03s]
```

3.3.2 Data visualization

➢ Data Imbalance visualization



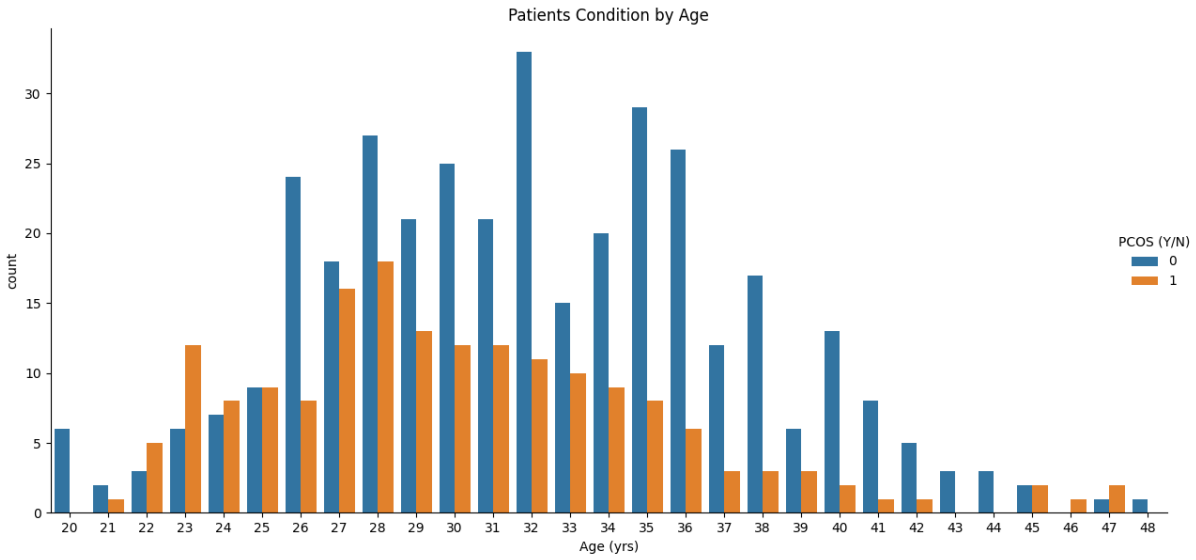
We used the following code to generate a 'countplot' of the number of women who have PCOS and number of women who do not to visualize the skew of the data. It is found that the data is 'Moderately Balanced'.

```
# Data Imbalance visualization plot
target = df['PCOS (Y/N)']
df.drop('PCOS (Y/N)', axis=1, inplace=True)
plt.figure(figsize=(8, 7))
sns.countplot(target)
plt.title('Data imbalance')
plt.show()

```

➢ Relationship between numerical variable and one or more categorical variables

Visualizing relationship between PCOS and age, blood group, number of marriage years:

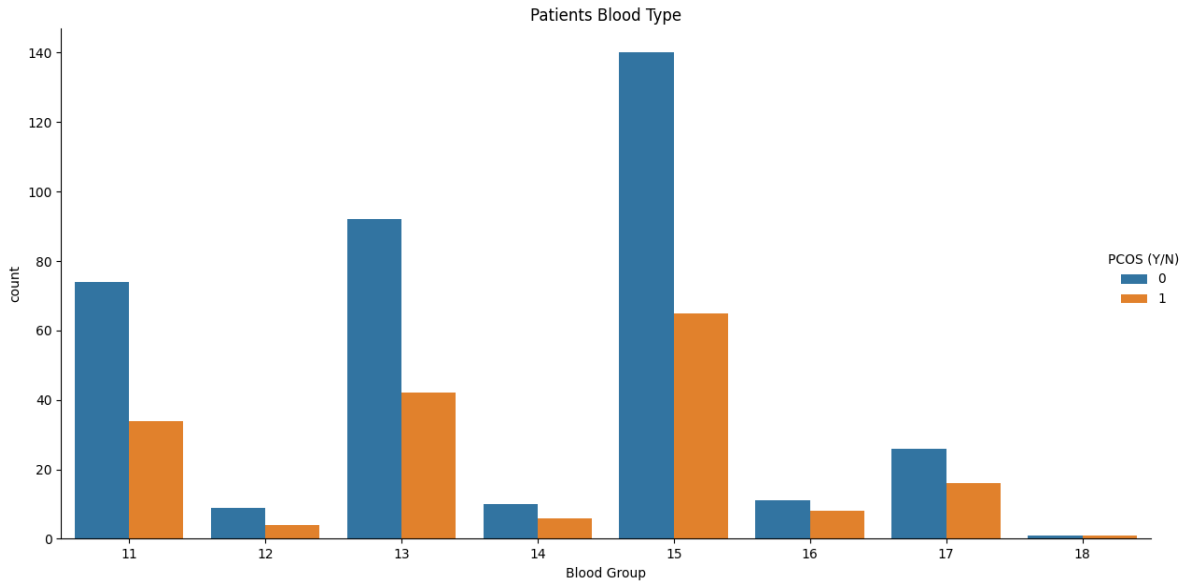
- Age and PCOS



FINDINGS: Patients aged 28 have the highest number of PCOS cases with 18 cases (3.3%). Patients aged 32 have the highest number of 'No PCOS' cases with a count of 33 cases (2.7%).

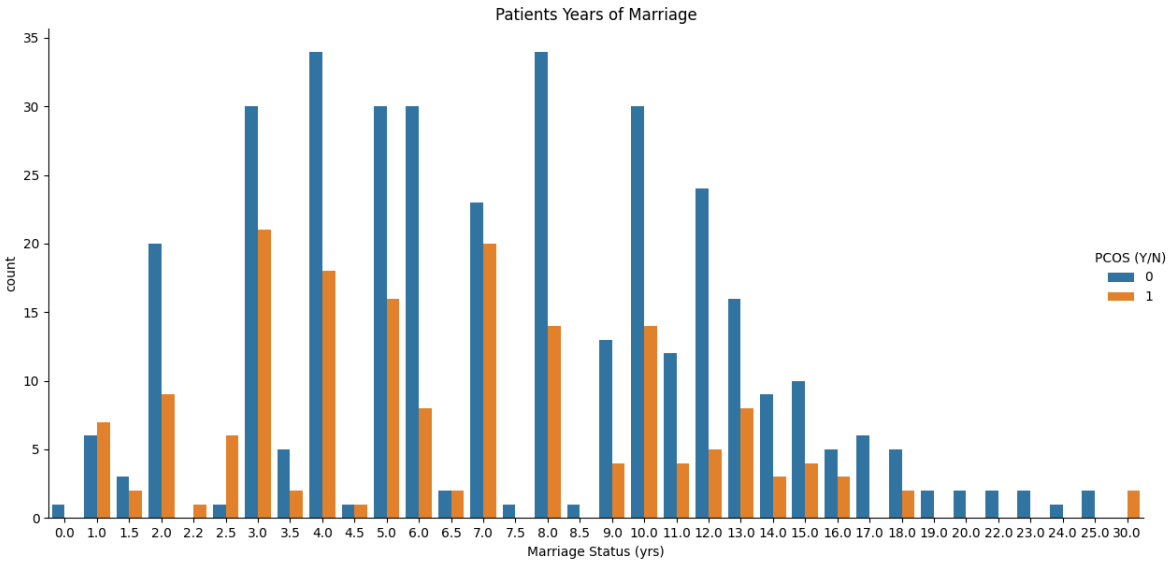
As age moves from 20 to 28, chances of PCOS increases and as it moves from 28 to menopause, the chances decrease at a relatively lower rate than the increase.

- **Blood Group and PCOS**



Most patients in this study have blood type O+ (15) with 205 patient observations. Least likely blood type is AB- (18) with 2 patient observations.

- Marriage years and PCOS



Patients with no PCOS, married 4 years have the highest number of observations.
Patients with PCOS, married 3 years have the highest number of observations.

As the number of married years increases, chances of the PCOS condition improving is expected as intercourse tends to have a positive impact on the condition.

Following code snippet was used to plot the above 3 graphs:

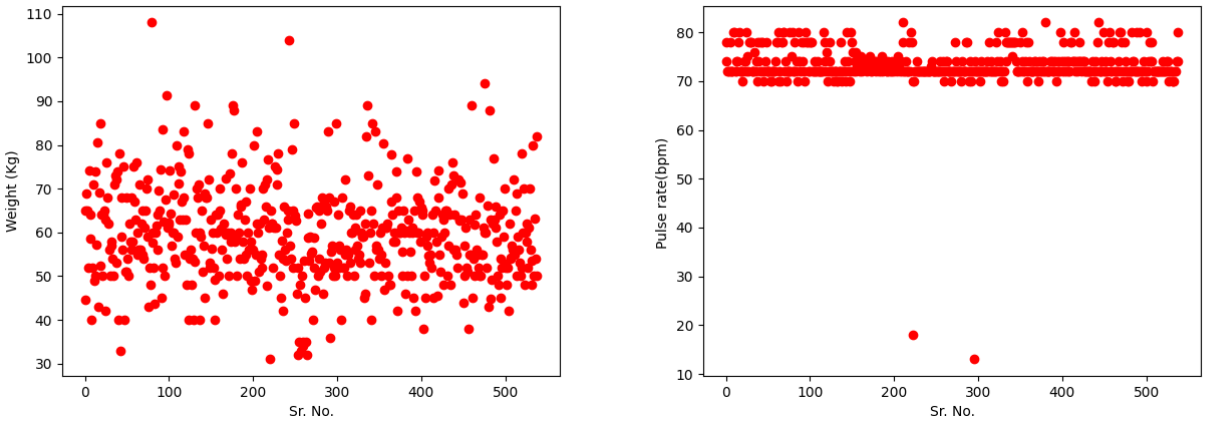
```
# Plots to show relationship between numerical variable and one or more categorical
variables

# Age and PCOS
age = sns.catplot(x='Age (yrs)', data=df, hue='PCOS (Y/N)', kind='count')
plt.title("Patients Condition by Age")
plt.show()

# Blood Group and PCOS
blood = sns.catplot(x='Blood Group', data=df, hue='PCOS (Y/N)', kind='count')
plt.title("Patients Blood Type")
plt.show()

# Marriage years and PCOS
marr_yrs = sns.catplot(x='Marriage Status (yrs)', data=df, hue='PCOS (Y/N)',
kind='count')
plt.title("Patients Years of Marriage")
plt.show()
```

3.3.3 Outlier analysis

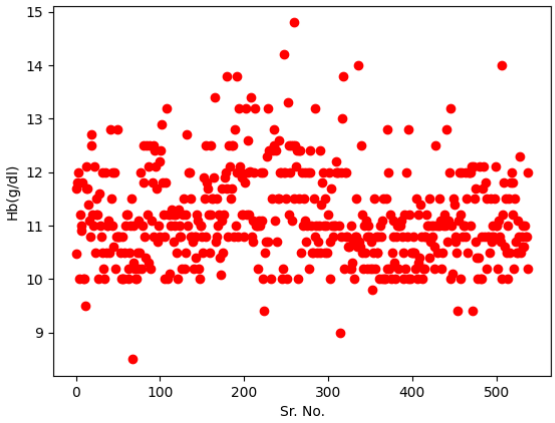
The following code was used to create the scatter plot for identifying outliers for all columns having non-binary values.

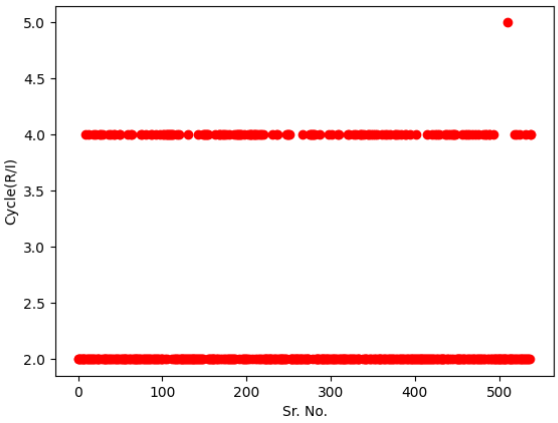
```
# Plotting for outlier detection
# Excluding columns which has binary values
cols = list(df.columns)

for col in cols:
    if(col.strip()[-5:]) != "(Y/N)":
        plt.scatter([var for var in range(len(df[col]))], df[col], color='red')
        plt.xlabel('Sr. No.')
        plt.ylabel(col)
        plt.show()
```

After the generation of all the plots, we looked at which ones indeed had outliers as some of the parameters like age, height, blood group, etc. had nothing to do with existence of outliers. Following are the scatter plots of the ones which we found to be relevant.

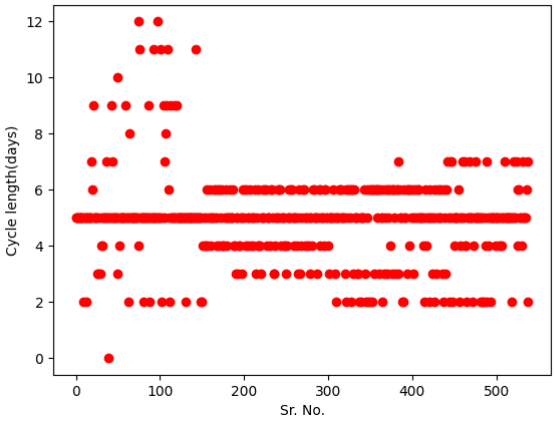


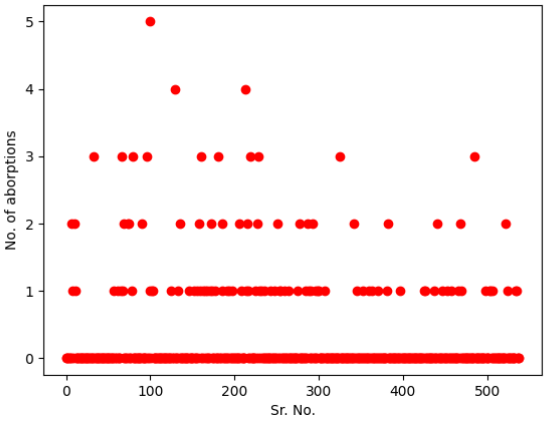
Two women appear to be extremely overweight (between 100-110 kgs) and two women have extremely low pulse-rate (between 10-20 bpm).



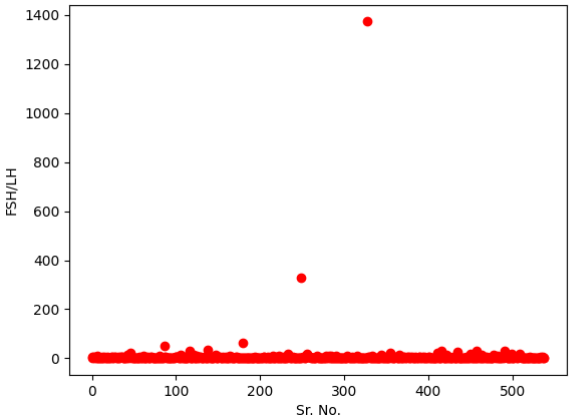


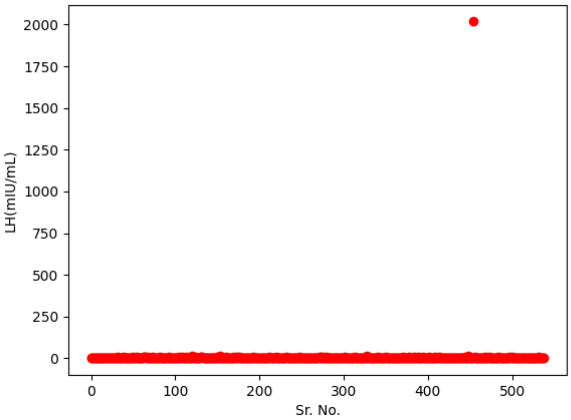
Hemoglobin is extremely high (around 15 g/dl) in one and extremely low (below 9g/dl) in another woman.



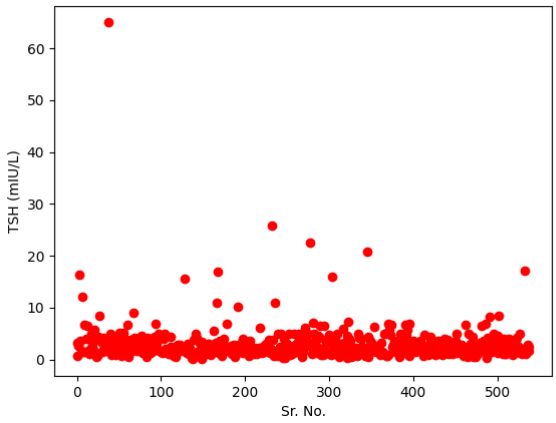


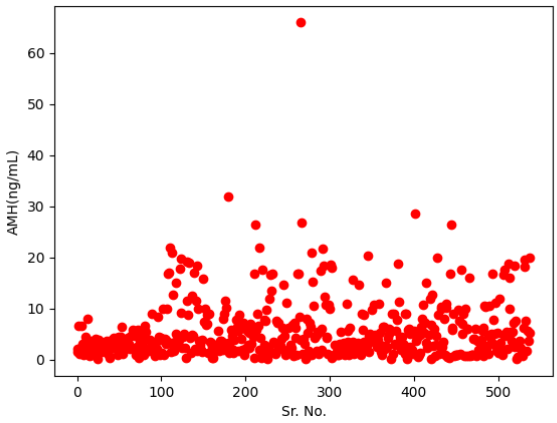
Cycle length is extremely fluctuate and number of abortions (miscarriages) are high in many women.



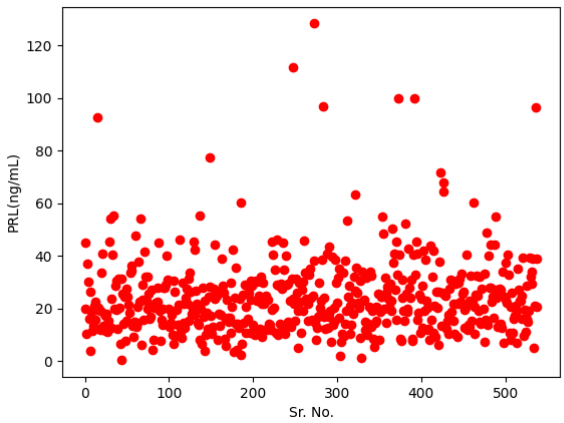


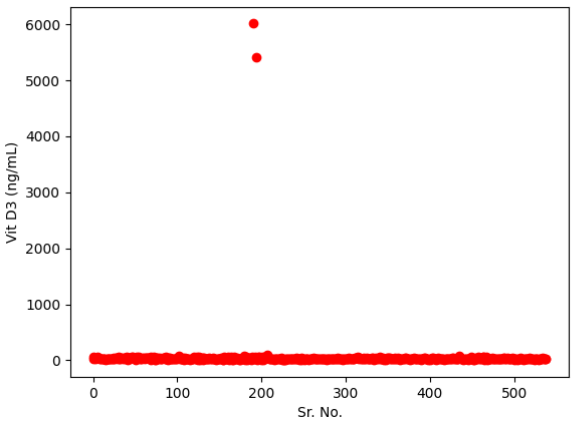
1 woman has an exceptionally high FSH level and 2 have relatively higher than the rest of the samples.. Only one woman has an extremely high LH level.

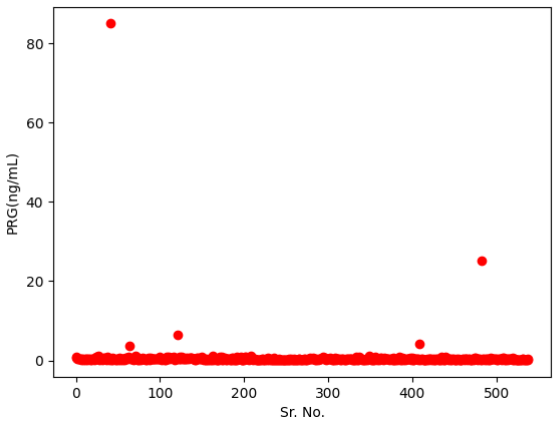


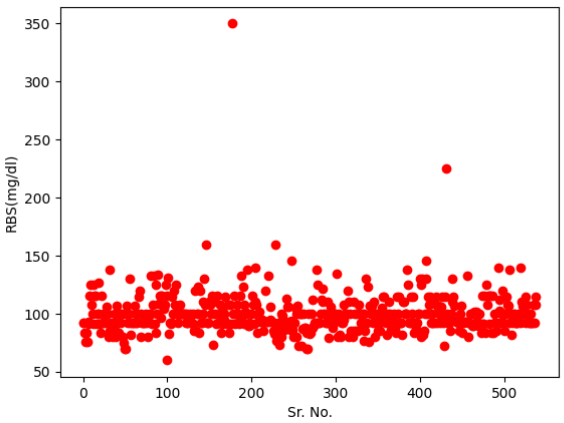


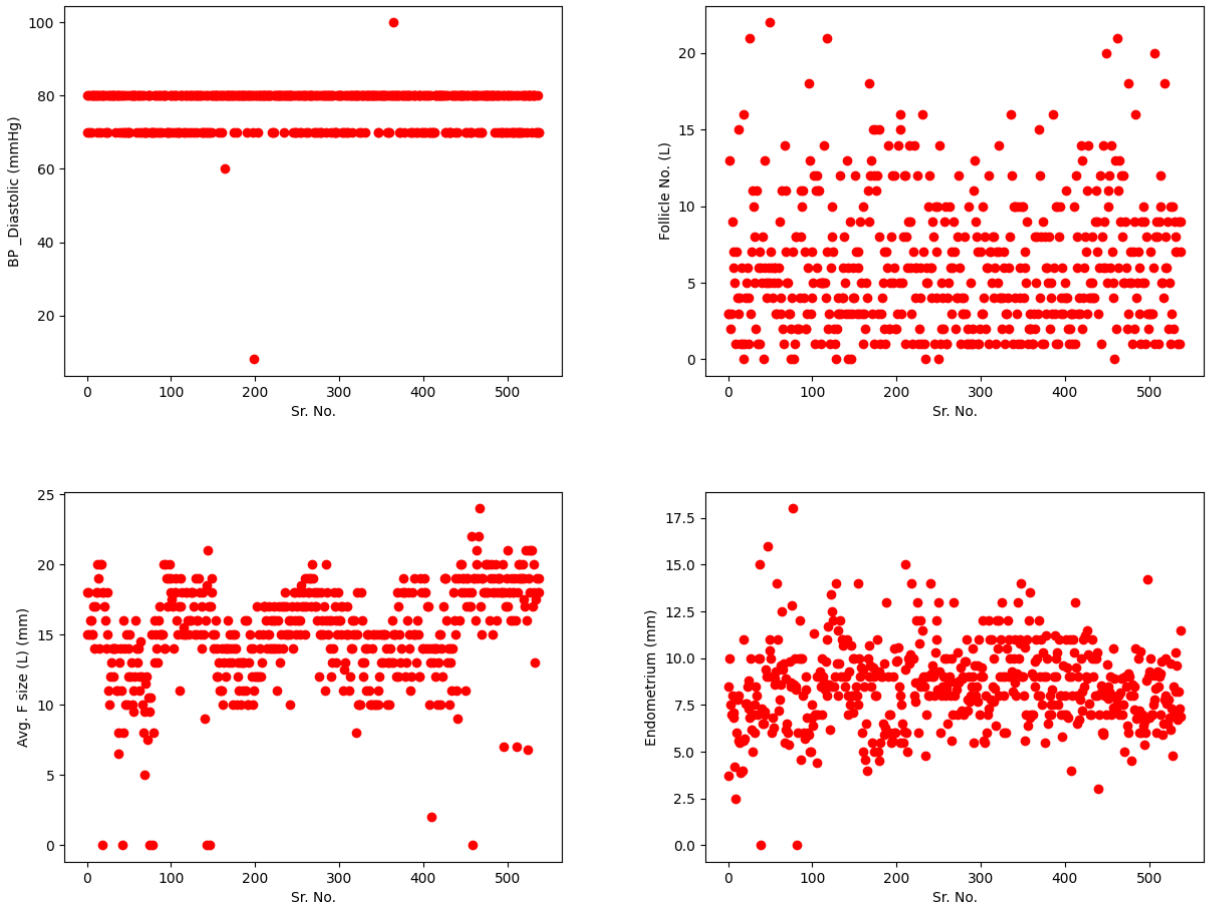
A few women have a high TSH and one woman has an extremely high AMH level.











Diastolic Blood pressure has two outliers with extremely low (at around 60mmHg and 5mmHg). Follicle numbers vary largely with a few women having extremely high numbers of follicles which would cause severely bad PCOS. Average follicle size lies between 5-20mm mostly, except for a few outliers. Endometrium sizes lie between 2.5-15mm with a few exceptional outliers.

3.4 Feature Selection

In machine learning, the data we are handling becomes very important and crucial, even more so than the model we use. This is the reason why we dedicated the entire section 3 to illustrate the various data transformation and clustering techniques. One aspect of handling data is feature selection and extraction, it answers the question 'How can we make the features most relevant for an efficient machine learning process?'.

Feature selection is the process of choosing the most relevant features from our data set to feed to our model. There is a distinct 'possibility' (this may depend and vary with model and other factors too) of a two fold advantage for doing so,

1. Training speed increases
2. Accuracy improves
3. Reduces overfitting

In essence, feature selection is a process, which may be done manually using domain experts or automatically by various techniques like:

A. Filter methods
B. Wrapper methods
C. Embedded methods
D. Hybrid methods

Having irrelevant or even partially relevant features can have a negative impact on the performance of the model. Feature extraction therefore has the primary aim of reducing the number of features in a dataset by creating new features from the existing one after which we discard the original features. [16] This is explained in detail in section 5.4.4. Now, we discuss some of the feature selection methods with relevant observations attached

3.4.1 Univariate selection

Filter methods pick up the intrinsic properties of the features measured via univariate statistics instead of cross-validation performance. These methods are faster and less computationally expensive than wrapper methods. The univariate selection technique which uses univariate statistical tests to determine and select those features which have the strongest relationship with the output variable. Each feature is compared to the target variable to determine if there are any statistically significant relationships between them, also referred to as analysis of variance. During these tests the code analyses the

relationship between one feature and target variable whilst ignoring the other features at a time. One such test is the “chi-squared statistical test for non-negative features” which we have employed to select the 10 best features from the PCOS prediction data set. [10]

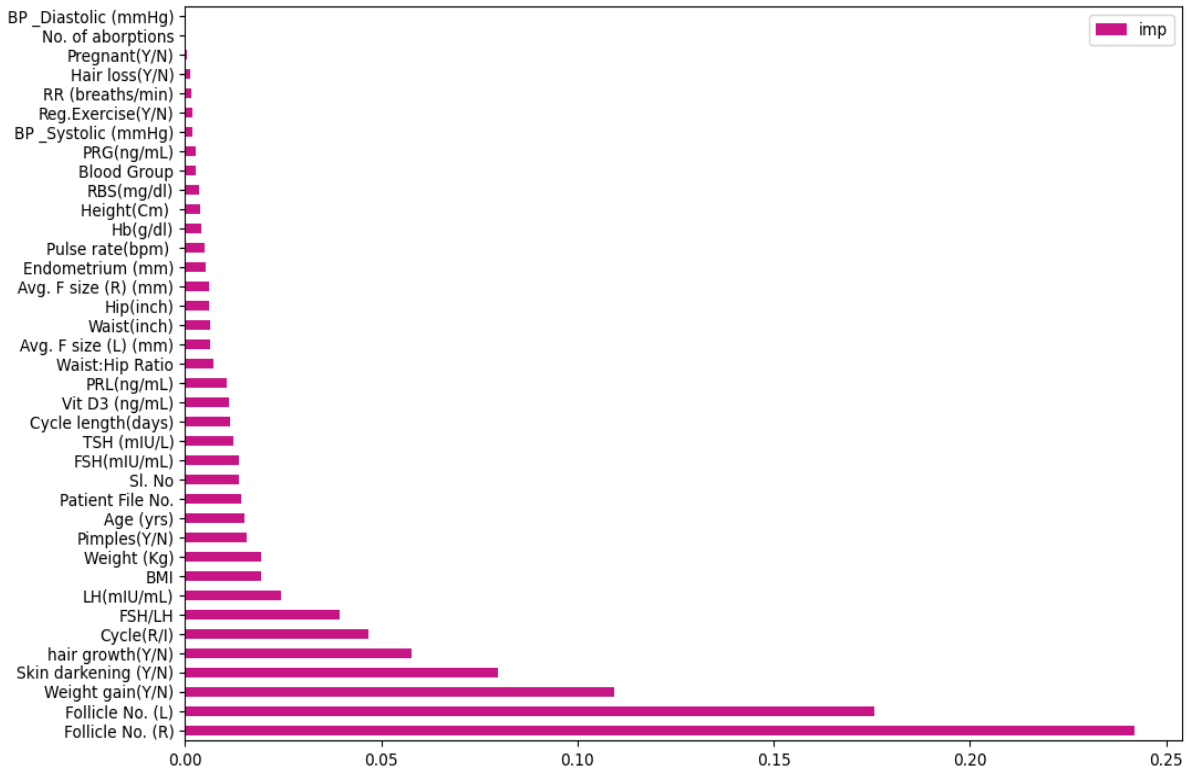
The scikit-learn library provides the SelectKBest class that can be used with a suite of different statistical tests to select a specific number of features.

```python
import pandas as pd
import numpy as np
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2

# Reading dataset
df = pd.read_csv('C:/Users/nicnnnn/Documents/PCOS_Project/Datasets/PCOS_withoutinfertility.csv')

# Treating errors in dataset
del df['AMH(ng/mL)']
del df['Unnamed: 42']  # All NaN value
del df['Marraige Status (Yrs)']  # 1 NaN value
del df['Fast food (Y/N)']  # 1 NaN value


# Dropping PCOS (Y/N) column and creating target column y for predictions
X = df.drop(columns=["PCOS (Y/N)"])
y = df["PCOS (Y/N)"]

# Applying SelectKBest class to extract top 10 best features
bestfeatures = SelectKBest(score_func=chi2, k=10)
fit = bestfeatures.fit(X,y)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(X.columns)

# Concatenating two dataframes for better visualization
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['Specs','Score']  #naming the dataframe columns
print(featureScores.nlargest(10,'Score'))  #print 10 best features
```

The following are the results of the univariate selection:

```
Python - UnivariateSelection.py:5 ✔

                  Specs        Score
22        Vit D3 (ng/mL)  9477.648952
15            LH(mIU/mL)  2558.471157
14           FSH(mIU/mL)  1601.145511
34      Follicle No. (R)   672.789402
33      Follicle No. (L)   573.647927
0                Sl. No   181.161504
16               FSH/LH    96.852985
27  Skin darkening (Y/N)    84.870716
26      hair growth(Y/N)    84.854623
25      Weight gain(Y/N)    65.554147
[Finished in 2.876s]
```

3.4.2 Feature Importance



As it can be seen that the Follicle No. has the highest importance, as high follicle numbers means PCOS.

The highest important symptom is weight gain (doctors suggest weight control through diet and exercise as the first and foremost step to be taken to cure PCOS).
Other symptoms like skin darkening, hair growth (at unwanted parts of the body) and FSH, LH levels have high importance too.

The code that was used to sort the features according to their respective importance ranks and plot them is:

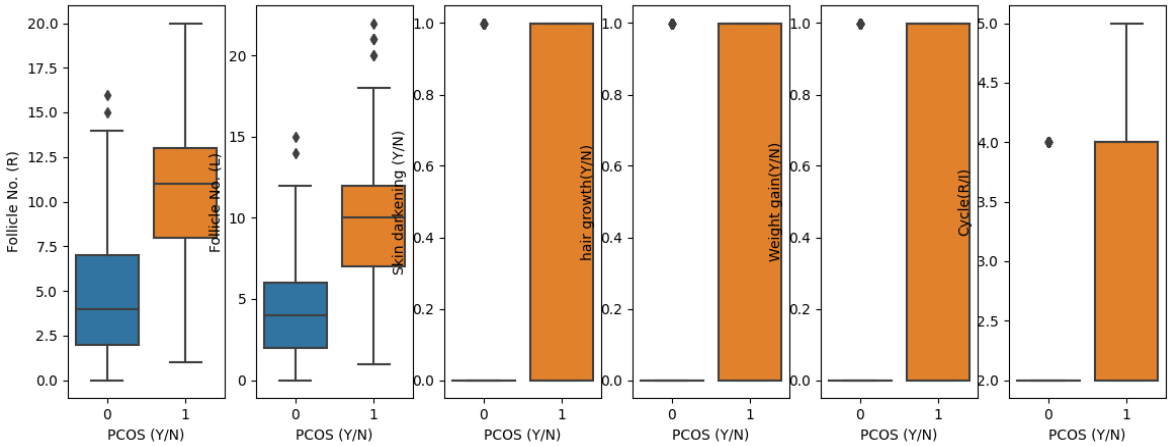
```
def get_fi(m, df):
    return pd.DataFrame({'col': df.columns, 'imp': m.feature_importances_}).sort_values('imp', ascending=False)


# lets get the feature importances for training set
fi = get_fi(rf, X_train)


def plot_fi(df):
    df.plot('col', 'imp', 'barh', figsize=(10, 10), color='mediumvioletred')


plot_fi(fi)

plt.show()
```

The 5 most important features

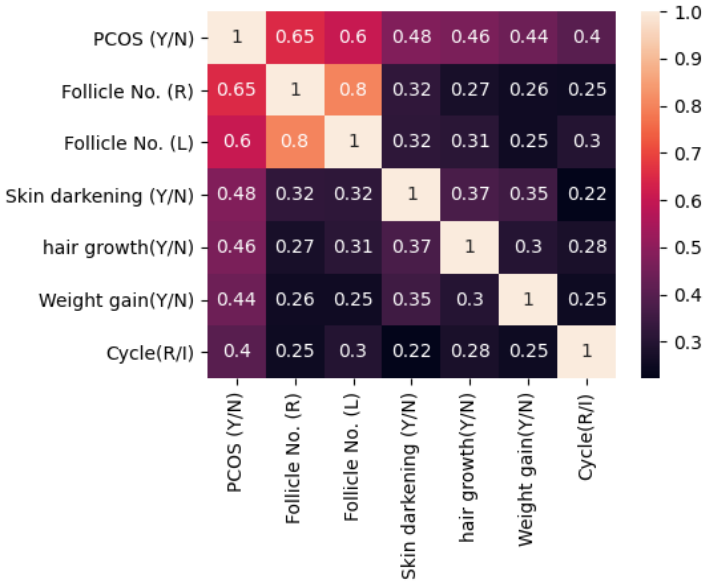
It can be clearly seen from the above feature selection and extraction techniques that the 5 most important features are:

1. 'Follicle No.(L)'
2. 'Skin darkening'
3. 'Hair growth'
4. 'Weight gain'
5. 'Cycle'

To visualize them better, we plotted the following boxplots:



To double check, we generated the correlation matrix for these 5 most important features:

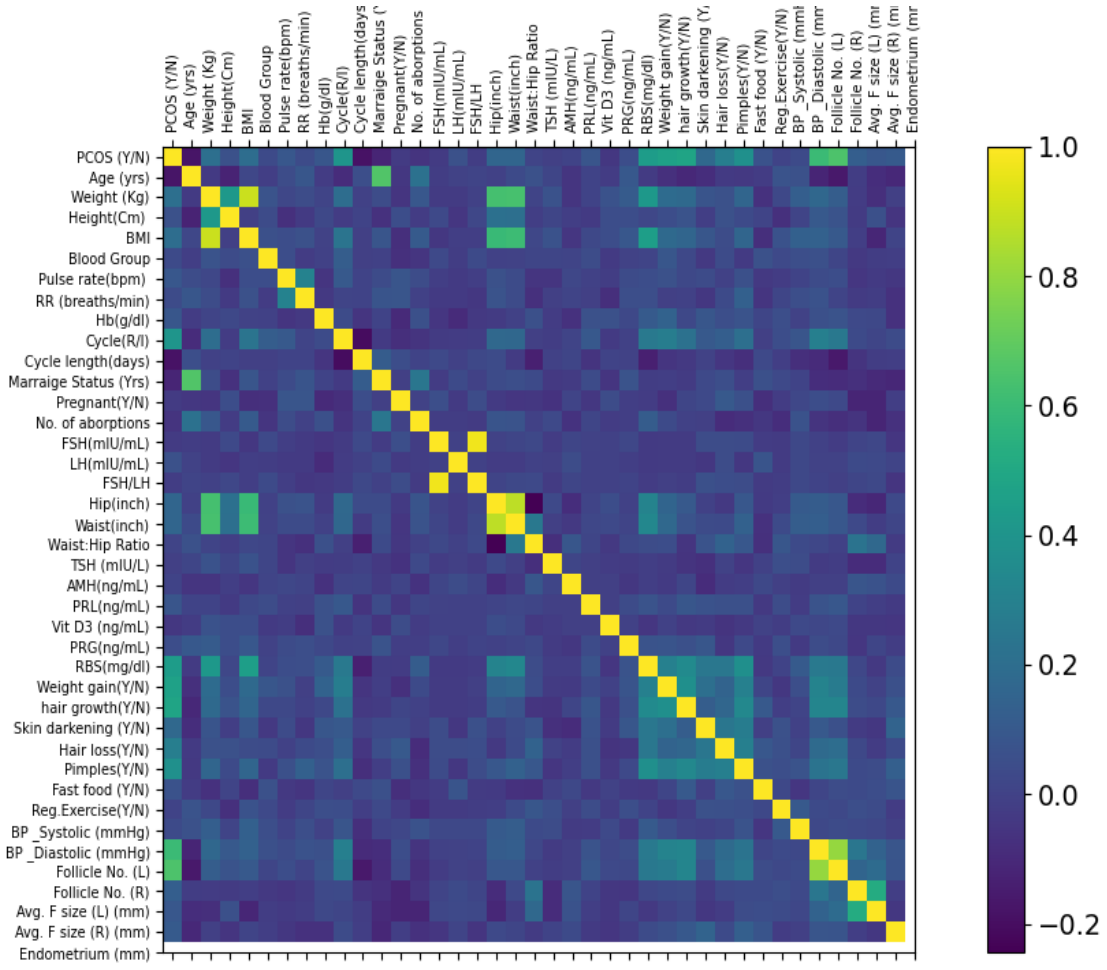


This correlation matrix corroborates our findings in the box plot
The following code was used to generate these:

```
plt.figure(figsize=(14,5))
plt.subplot(1,6,1)
sns.boxplot(x='PCOS (Y/N)',y='Follicle No. (R)',data=df_noinf)
plt.subplot(1,6,2)
sns.boxplot(x='PCOS (Y/N)',y='Follicle No. (L)',data=df_noinf)
plt.subplot(1,6,3)
sns.boxplot(x='PCOS (Y/N)',y='Skin darkening (Y/N)',data=df_noinf)
plt.subplot(1,6,4)
sns.boxplot(x='PCOS (Y/N)',y='hair growth(Y/N)',data=df_noinf)
plt.subplot(1,6,5)
sns.boxplot(x='PCOS (Y/N)',y='Weight gain(Y/N)',data=df_noinf)
plt.subplot(1,6,6)
sns.boxplot(x='PCOS (Y/N)',y='Cycle(R/I)',data=df_noinf)

plt.show()

#correlation matrix heatmap
plt.figure(figsize=(6,5))
sns.heatmap(df_noinf.corr(), annot=True)
plt.show()
```

3.4.3 Correlation matrix with heatmap

- Colour-encoded matrix of all parameters



- numbers closer to -1 defining a strong correlation,

- numbers closer to 1 defining a strong positive correlation and
- numbers closer to 0 meaning little to no correlation.

To get a graphical understanding of how different attributes are related with one another, we use the above generated colour-encoded correlation matrix.

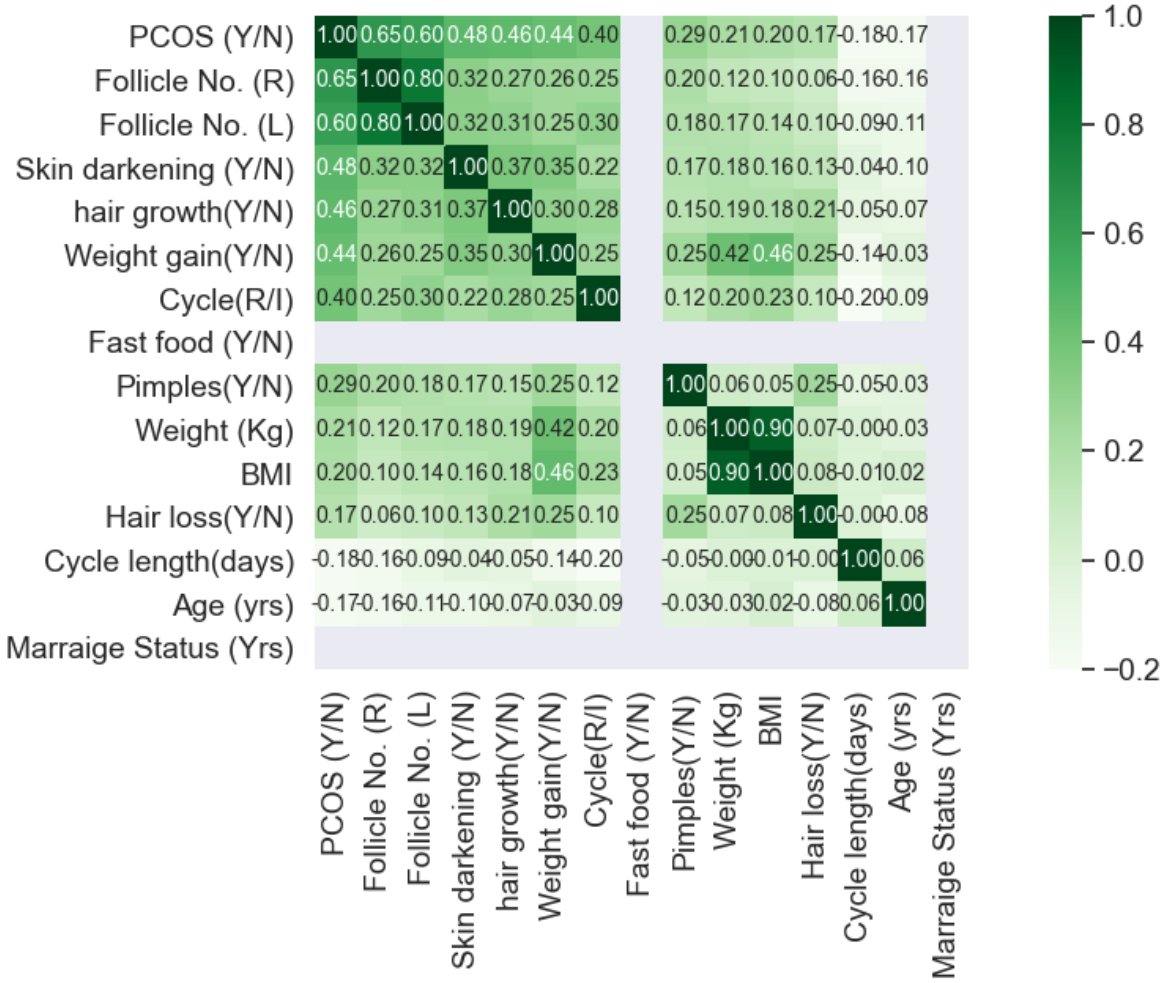
There are some vital observations that can be made by just visualizing this graph:

- Features 'Hip(inch)' or 'Waist(inch)' are highly interrelated with each other and the features 'Weight(kg)' and 'BMI'. Women having higher weights obviously have greater hip and waist circumference lengths. Furthermore, women who have higher weights are more likely to develop 'PCOS (Y/N)', as can be seen.
- Features 'BP _Diastatic (mmHg)' and 'Follicle No.' are highly related to 'PCOS (Y/N)', highlighting the crucial role this feature plays in detecting PCOS. Follicle numbers are obviously high in women having PCOS.
- Symptoms such a 'Weight gain (Y/N)', 'hair growth (Y/N)', 'Skin darkening (Y/N)', 'Hair loss (Y/N)' and 'Pimples (Y/N)' along with 'RBS(mg/dl)' are in the moderate region of interrelationship (0.5-0.6) with 'PCOS (Y/N)'

The following code was used to plot the above correlation matrix.

```
40  # Plotting Correlation Matrix
41  f = plt.figure(figsize=(19, 15))
42  plt.matshow(df.corr(), fignum=f.number)
43  plt.xticks(range(df.shape[1]), df.columns, fontsize=7, rotation=90)
44  plt.yticks(range(df.shape[1]), df.columns, fontsize=7)
45  cb = plt.colorbar()
46  cb.ax.tick_params(labelsize=14)
47  plt.show()
48
```

- Weighted Correlation Heatmap of certain important symptoms



The above weighted heatmap was generated using the following code:

```
# Weighted Correlation Heatmap
corrmat = df.corr()
corrmat["PCOS (Y/N)"].sort_values(ascending=False)
plt.figure(figsize=(12, 12))
k = 12  # number of variables with positive for heatmap
l = 3  # number of variables with negative for heatmap
cols_p = corrmat.nlargest(k, "PCOS (Y/N)")["PCOS (Y/N)"].index
cols_n = corrmat.nsmallest(l, "PCOS (Y/N)")["PCOS (Y/N)"].index
cols = cols_p.append(cols_n)

cm = np.corrcoef(df[cols].values.T)
sns.set(font_scale=1.25)
hm = sns.heatmap(cm, cbar=True, cmap="Greens", annot=True, square=True, fmt='.2f',
                 annot_kws={'size': 10}, yticklabels=cols.values, xticklabels=cols.values)
plt.show()
```

3.5 Feature Extraction

Feature extraction provides an alternative approach to data preparation for tabular data, where all data transforms are applied in parallel to raw input data and combined together to create one large dataset. We have discussed however in the previous sections of the problems with a very large number of features that can result in computational issues and affect the algorithmic performance, and hence the prediction of outcome itself if they are not independent. Ideally with feature extraction a expert driven approach (who says which features are important) must be complemented with the data-driven approach.

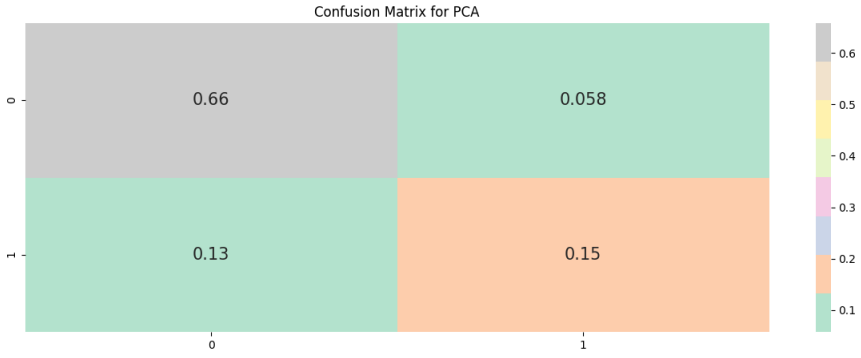
Say original data set has a large number of features, n
The feature extraction technique should suggest a "subset of features" that are important
The new subset = m features, where m < n

In feature extraction when creating a new subset with m < n, the physical properties of each feature is lost. We are left with just numerical quantification. However, they are known to give good performance on algorithms when applied to this new subset. [14] The linear transformation technique called the Principle Component Analysis is one of the most widely used.

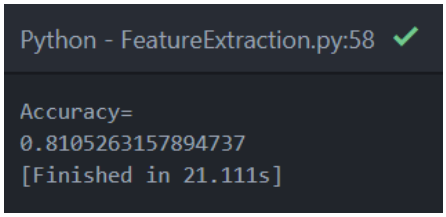
3.5.1 Principle Component Analysis

The PCA is able to maximize the variances and minimize the reconstruction error by projecting original data into a set of orthogonal axes and each getting ranked in the order of importance. In other words, PCS takes data input and finds a combination of input features that best summarize the original data distribution.

In our case, we first perform PCA in the whole dataset to reduce our data to just two dimensions and we will then construct a data frame with our new features and their respective labels. We run this on a random forest classifier algorithm to demonstrate, the following is the obtained confusion matrix



The screenshot attached below shows the output.



The code used is shown below. It is to be noted that this is done just for illustration purposes, a more detailed explanation of Random Forest is given in section 5.

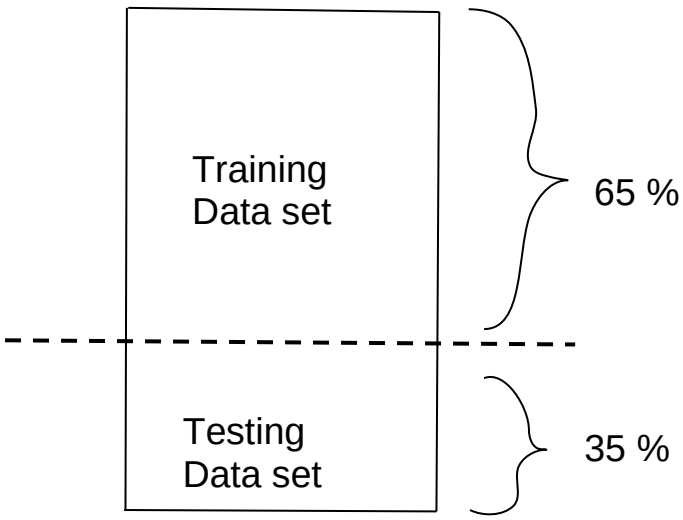
```
# Performing standard scalar normalization
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Applying PCA
pca = PCA()
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)

explained_variance = pca.explained_variance_ratio_
pca = PCA(n_components=1)
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)

# RandomForestClassifier use
classifier = RandomForestClassifier(max_depth=2, random_state=0)
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

# Results and Confusion matrix
print('Accuracy=')
print(accuracy_score(y_test, y_pred))
plt.subplots(figsize=(15, 5))
cf_matrix= confusion_matrix(y_test, y_pred)
sns.heatmap(cf_matrix/np.sum(cf_matrix), annot=True, annot_kws={'size': 15}, cmap='Pastel2')
plt.title("Confusion Matrix for PCA")
plt.show()

pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
PCA_df = pd.DataFrame(data = X_pca, columns = ['PC1', 'PC2'])
PCA_df = pd.concat([PCA_df, df['class']], axis = 1)
PCA_df['class'] = LabelEncoder().fit_transform(PCA_df['class'])
PCA_df.head()
```

4. Training the Model

4.1 Classification & splitting of data

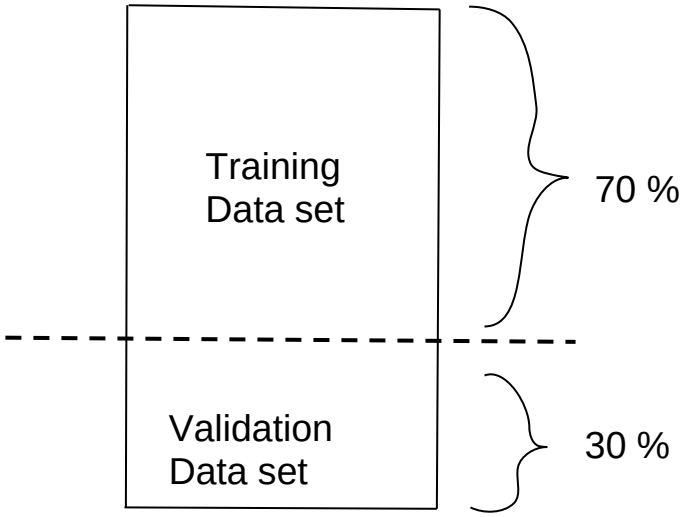
Within the supervised learning process based on the nature of the job identified after data exploration we either do regression or classification. In our case we do classification (as the target output is the prediction of PCOS which has a discrete number of values that it can take). The motive is to learn from already labeled data on how to predict the class of unlabeled data. In other words, classification is a discrete form of supervised learning - since in our case there are just 2 categories (YES a person has PCOS/ NO a person doesn't have PCOS), it is a case of a **Binary Classification problem**. Regression analysis is used when there are continuous variables to be predicted.

So, building on this idea our machine learning model should have a training set and a testing set. The training set also includes the validation set. The Machine learns from or gains experience from the training data. We then test those gained experience or properties with another data set to comment on the accuracy of a particular model under study, which happens to be the testing data set.



Before we use the data for various algorithms to do a comparative study, we split the data into training and testing in the proportion of 65 % and 35 %. We can use scikit-learn's train_test_split function to split our data in python.

Furthermore, during the generation of the ROC curve, we splitted the training data set into 30% for validation and 70% for training as follows.



Analysing or testing of various classifiers is necessary to gauge an understanding of which model works best for our data set and research. This is done with the help of various techniques and statistical methods. Some of the common techniques which we use in our mid sem report include confusion matrix, ROC curve, precision value, recall value and F1 score. We aim to develop this further for the end term report with the end goal being to analyse these values for a number of algorithms and then selecting the best classifier algorithm. Therefore, now we attempt to employ various classifiers discussed till mid-sem in section 4. 3. But before that, we have detailed explanations of the various terminologies.

4.2 Confusion Matrix

The model we are developing is for a binary classification problem. It predicts if the patient has PCOS (Yes or + ve) or does not have PCOS (NO or - ve). Now this matrix gives us an understanding of True positive, True Negative, False positive, False negative values. These terms mean,

- True positive - Number of samples which are predicted to have PCOS and actually have PCOS. (correct prediction)
- True negative - Number of samples which are predicted to not have PCOS and actually does not have PCOS. (correct prediction)
- False positive - Number of samples which are predicted to have PCOS and actually does not have PCOS. (incorrect prediction)
- False negative - Number of samples which are predicted to not have PCOS and actually have PCOS. (incorrect prediction) [7]

When applying a classifier algorithm to a data set (testing sample) we first obtain the confusion matrix with these four terms of TP, TN, FP, FN. Amongst these four terms, they have different costs and benefits or in other words, risks and gains when analysing a particular classification model. For instance, an incorrect prediction of a PCOS patient as a non-PCOS patient has higher risk since this involves valuable time lost in the treatment of the disease.

So using these 4 values we develop accuracy, precision recall and F1 score all of which is elaborated in sections after this. We therefore, look with great interest at these parameters that we obtain from the confusion matrix for different algorithms or classification models to determine the best model as we develop this project further for the end-term paper.

4.3 ROC (Receiver Operating Characteristic) Curve

The confusion matrix corresponds to a single point on the ROC curve which is a graph drawn between tp rate and fp rate. The ROC graph itself is a 2-D plot of the tp-rate or sensitivity on the y axis and fp-rate or specificity on the x axis.
Although accuracy is a good enough measure in our case, as we have noticed the data set we are using is not a highly skewed data set, we will still be calculating the tp rate and fp rate to give a comprehensive picture and build confidence of the model.

The optimum classifier (not achievable) will have the fp rate to be 0 and tp rate at 1 so all the patients who have PCOS are rightly predicted and no patient who doesn't have PCOS is wrongly predicted to have it. There are always trade-offs between these two. In general we say the curve with higher AUC (Area Under Curve) is better, AUC > 0.5. [7]

4. 4 Performance metrics

- **Precision:** Referred to as precision in the information retrieval domain, this is the ability of the model not to classify a sample positive that is actually negative. For each class (0 or 1), it is defined as true positives divided by the sum of true positives and false positives. This is the fraction of predicted positives that are actually positive. Therefore,

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Recall or tp rate:** Referred to as recall in the information retrieval domain, or else known as tp rate is the ability of the model to find all the positives. It is defined, for each class, as the ratio of true positives and the sum of true positives and false

negatives. Also known as 'sensitivity' as this is the fraction of actual positives that are predicted positive. Therefore,

$$\text{Recall} = \frac{TP}{TP + FN}$$

- **F1 score:** This is the harmonic mean of precision and recall. We are interested in high detection rates of positives, so we want TP to be high, that is, recall to be high. Therefore, we want a large data set for high detection rates but that would mean precision would reduce. So there is a tie between precision and recall. F1 score is hence a measure of test's accuracy where 1 is its best score and 0 is its worst. It is helpful for comparing two classifiers. Therefore,

$$\text{F1 score} = \frac{2 * precision * recall}{precision + recall}$$

- **fp rate:** Sometimes referred to as 'fall-out' or 'false alarm', this is the proportion of negative cases incorrectly identified as positive cases. Therefore,

$$\text{fp rate} = \frac{FP}{TN + FP}$$

- **Accuracy:** Accuracy is a measure of the proportion of correct predictions for the test data. Also referred to as success rate. Therefore,

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$$

- **Misclassification rate :** This is the measure of the proportion of wrong predictions for the test data. Also found using (1 - success rate). Therefore,

$$\text{Misclassification rate} = \frac{FP + FN}{TP + FP + TN + FN}$$

5. Classifiers

5.1 Naive Bayes Classifier

Naive Bayes algorithm is one of the oldest forms of machine learning. This algorithm is, as the name suggests, based on the Bayes Theorem.

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

The naive Bayes classifier is a non-parametric method which uses the Bayes theorem as a model to estimate the priors and likelihoods of an unseen sample from the given dataset. This classifier is however used with categorical features. When we have data which is continuous in nature, then we apply "**Gaussian Naive Bayes**". Here we make a gross assumption that every independent feature behaves like a gaussian probability density function. The parameters of the gaussian function are obtained from the data sample itself.

5.1.1 Naive Bayes Model Building

The Gaussian Naive Bayes model with all features considered was built as follows:

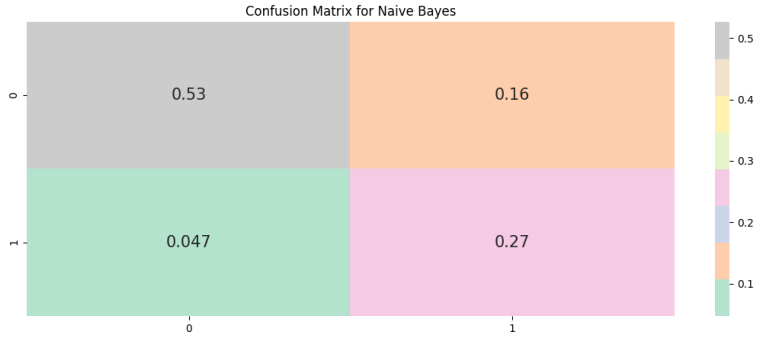
1. Splitting training and testing set with test size as 0.35
2. Creating Logistic Regression model and fitting our train and test sets in it
3. Predicting output using test set
4. Model evaluation - finding score in test data, number of right and wrong classifications.
5. Finding accuracy and other measures like Mean Absolute Error, Mean Squared Error, Root Mean Squared Error
6. Classification report generated
7. Confusion matrix plotted
8. ROC curve was plotted and AUC found

The Gaussian Naive Bayes model by selecting features considered was built as follows:

1. Selecting features with correlation >0.40 (5 features- 'Follicle No.(L)', 'Skin darkening', 'Hair growth', 'Weight gain', 'Cycle' satisfied the condition)
2. Splitting training and testing set with test size as 0.35
3. Creating Logistic Regression model and fitting our train and test sets in it
4. Predicting output using test set
5. Model evaluation - finding score in test data, number of right and wrong classifications.
6. Finding accuracy and other measures like Mean Absolute Error, Mean Squared Error, Root Mean Squared Error
7. Classification report generated

8. Confusion matrix plotted
9. ROC curve was plotted and AUC found

5.1.2 Performance measures for Naive Bayes with all features considered



Accuracy = $\frac{0.53+0.27}{0.16+0.53+0.27+0.047}$ = 0.7944 , i.e. 79.44%

Misclassification error = 1 - 0.7944 = 0.2055 , i.e. 20.55%

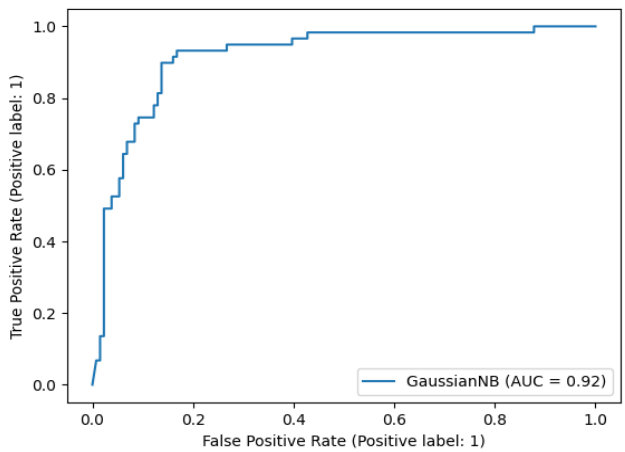
Precision = $\frac{0.27}{0.27+0.16}$ = 0.6279

Recall or tp rate = $\frac{0.27}{0.27+0.047}$ = 0.8517

fp rate = $\frac{0.16}{0.16+0.53}$ = 0.2318

F1 score = $\frac{2*0.6279*0.8517}{0.6279+0.8517}$ = 0.7228 ~ 0.72

ROC curve for Gaussian Naive Bayes model with all features considered-
As it can be seen from plot, Area under Curve=0.92



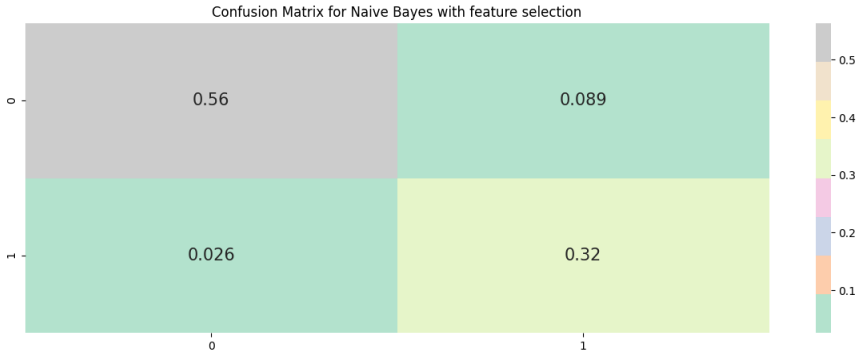
The following code was used for the above:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn import metrics
import pickle
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

# Reading dataset
df = pd.read_csv('C:/Users/nicnnnn/Documents/PCOS_Project/Datasets/PCOS_withoutinfertility.csv')

# Treating errors in dataset
del df['AMH(ng/mL)']
del df['Unnamed: 42']  # All NaN value
del df['Marraige Status (Yrs)']  # 1 NaN value
del df['Fast food (Y/N)']  # 1 NaN value

# Dropping PCOS (Y/N) column and creating target column y for predictions
X = df.drop(columns=["PCOS (Y/N)"])
y = df["PCOS (Y/N)"]

# Splitting data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.35)

# Creating NaiveBayes Classifier
gnb = GaussianNB()
gnb.fit(X_train, y_train)

# Prediction
y_pred=gnb.predict(X_test)

# Model Evaluation
print(f"Score in Test Data : {gnb.score(X_test,y_test)}")

cm=confusion_matrix(y_test, y_pred)
p_right=cm[0][0]+cm[1][1]
p_wrong=cm[0][1]+cm[1][0]

print(f"Right classification : {p_right}")
print(f"Wrong classification : {p_wrong}")

# Finding Naive Bayes accuracy and other measures
print("Naive Bayes model accuracy(in %):", gnb.score(X_test, y_test)*100)
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))

# Generating Classification Report for NaiveBayesClassifier
classi_report = classification_report(y_test, y_pred)
print(classi_report)

# Plotting the confusion matrix for NaiveBayesClassifier
plt.subplots(figsize=(15, 5))
cf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(cf_matrix/np.sum(cf_matrix), annot=True, annot_kws={'size': 15}, cmap='Pastel2')
plt.title("Confusion Matrix for Naive Bayes")
plt.show()

# Plotting ROC curve with AUC
metrics.plot_roc_curve(gnb, X_test, y_test)
plt.show()
```

'Classification Report' obtained as code output for the above Naive Bayes without feature selection code:

```
Python - NaiveBayesModel.py:31 ✔

Naive Bayes model accuracy(in %): 79.47368421052632
Mean Absolute Error: 0.20526315789473684
Mean Squared Error: 0.20526315789473684
Root Mean Squared Error: 0.45305977298225986
              precision    recall  f1-score   support

           0       0.92      0.77      0.84       130
           1       0.63      0.85      0.72        60

    accuracy                           0.79       190
   macro avg       0.77      0.81      0.78       190
weighted avg       0.83      0.79      0.80       190
```

5.1.3 Performance measures for Naive Bayes after feature selection



Accuracy = $\frac{0.56+0.32}{0.056+0.32+0.026+0.089}$ = 0.8844 , i.e. 88.44%

Misclassification error = 1 - 0.8844 = 0.1156 , i.e. 11.56%

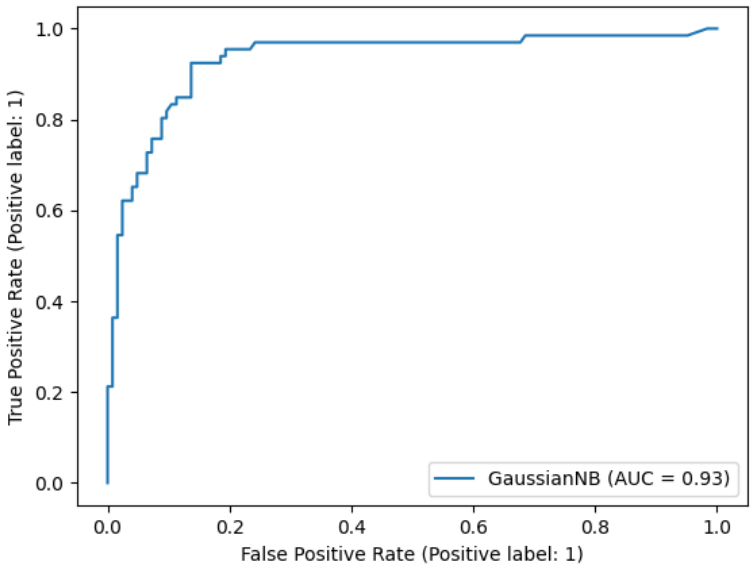
Precision = $\frac{0.32}{0.32+0.089}$ = 0.7823

Recall or tp rate = $\frac{0.32}{0.32+0.026}$ = 0.9248

fp rate = $\frac{0.089}{0.089+0.56}$ = 0.1371

F1 score = $\frac{2*0.7823*0.9248}{0.7823+0.9248}$ = 0.8476

ROC curve for Gaussian Naive Bayes model after feature selection-
As it can be seen from plot, Area under Curve=0.93



The following additional feature selection code with the few changes to previous code was used for the above:

```
# Reading datasets
df_inf = pd.read_csv('C:/Users/nicnnnn/Documents/PCOS_Project/Datasets/PCOS_infertility.csv')
df_noinf = pd.read_csv('C:/Users/nicnnnn/Documents/PCOS_Project/Datasets/PCOS_withoutinfertility.csv')

# Sampling data from df_inf and df_noinf
df_inf.sample(5)
df_noinf.sample(5)

## Feature Selection
# Identifying Features which have more than 0.40 correlation with PCOS(Y/N)
corr_features=df_noinf.corrwith(df_noinf["PCOS (Y/N)"]).abs().sort_values(ascending=False)
# features with correlation more than 0.4
corr_features=corr_features[corr_features>0.4].index
corr_features
df_inf.corrwith(df_inf["PCOS (Y/N)"]).abs()
df_noinf=df_noinf[corr_features]
df_noinf.head()
df_noinf.columns

# Dropping PCOS (Y/N) column and creating target column y for predictions
X = df_noinf.drop(columns=["PCOS (Y/N)"])
y = df_noinf["PCOS (Y/N)"]
```

'Classification Report' obtained as code output for the Naive Bayes with feature selection code:

```
Python - NaiveBayesModel_withFeatureSelection.py:11 ✔

Score in Test Data : 0.8842105263157894
Right classification : 168
Wrong classification : 22
Naive Bayes with feature selection model accuracy(in %): 88.42105263157895

Mean Absolute Error: 0.11578947368421053
Mean Squared Error: 0.11578947368421053
Root Mean Squared Error: 0.3402785236893603
              precision    recall  f1-score   support

           0       0.96      0.86      0.91       124
           1       0.78      0.92      0.85        66

    accuracy                           0.88       190
   macro avg       0.87      0.89      0.88       190
weighted avg       0.90      0.88      0.89       190

[Finished in 35.887s]
```

5.2 k-Nearest Neighbours Classifier

kNN is an Instance-Based Learning that compares new samples with those samples stored in memory at the time of training of the dataset. A new sample is classified by measuring its distances with the samples retrieved from memory, defined in terms of standard Euclidean Geometry, that is, distance between points in n dimensional space. The accuracy of this model depends upon two factors:

- The value of 'K'
- The number of selected features.

5.2.1 kNN Model Building

The k Nearest Neighbours model with all features considered was built as follows:

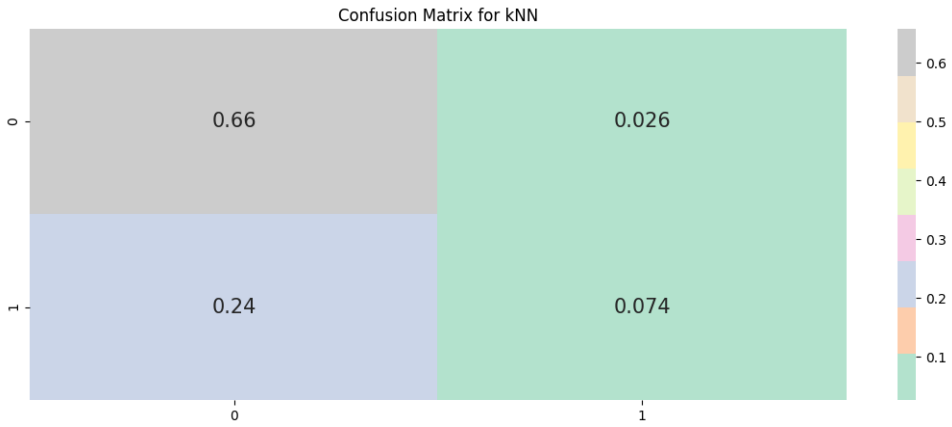
1. Splitting training and testing set with test size as 0.35
2. Creating Logistic Regression model and fitting our train and test sets in it
3. Predicting output using test set
4. Model evaluation - finding score in test data, number of right and wrong classifications.
5. Finding accuracy and other measures like Mean Absolute Error, Mean Squared Error, Root Mean Squared Error
6. Classification report generated
7. Confusion matrix plotted

8. ROC curve was plotted and AUC found

The kNN model by selecting features considered was built as follows:

1. Selecting features with correlation >0.40 (5 features- 'Follicle No.(L)', 'Skin darkening', 'Hair growth', 'Weight gain', 'Cycle' satisfied the condition)
2. Splitting training and testing set with test size as 0.35
3. Creating Logistic Regression model and fitting our train and test sets in it
4. Predicting output using test set
5. Model evaluation - finding score in test data, number of right and wrong classifications.
6. Finding accuracy and other measures like Mean Absolute Error, Mean Squared Error, Root Mean Squared Error
7. Classification report generated
8. Confusion matrix plotted
9. ROC curve was plotted and AUC found

5.2.2 Performance measures for kNN with all features considered



Accuracy = $\frac{0.074+0.66}{0.074+0.66+0.026+0.24}$ = 0.734 , i.e. 73.4%

Misclassification error = 1 - 0.734 = 0.266 , i.e. 26.6 %

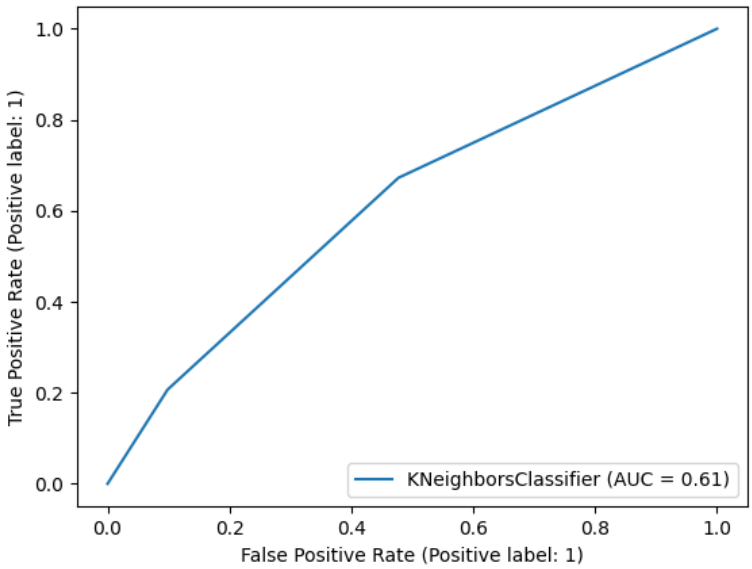
Precision = $\frac{0.074}{0.074+0.026}$ = 0.7400

Recall or tp rate = $\frac{0.074}{0.074+0.24}$ = 0.2356

fp rate = $\frac{0.026}{0.026+0.66}$ = 0.0379

F1 score = $\frac{2*0.74*0.2356}{0.74+0.2356} = 0.3574$

ROC curve for kNN model with all features considered-
As it can be seen from plot, Area under Curve=0.61



The following code was used for the above:

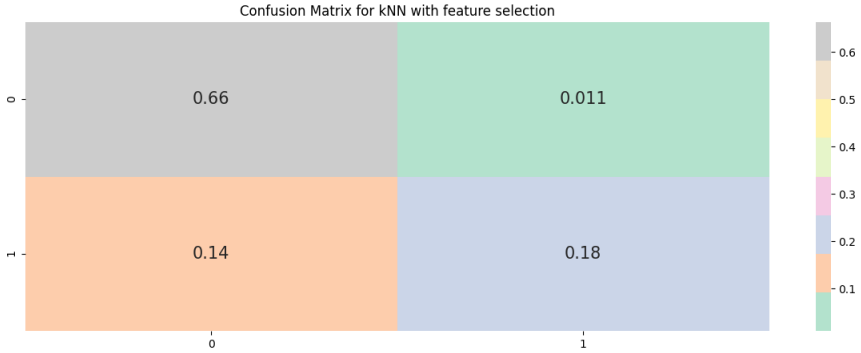
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
import pickle
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

# Reading dataset
df = pd.read_csv('C:/Users/nicnnnn/Documents/PCOS_Project/Datasets/PCOS_withoutinfertility.csv')

# Treating errors in dataset
del df['AMH(ng/mL)']
del df['Unnamed: 42']  # All NaN value
del df['Marraige Status (Yrs)']  # 1 NaN value
del df['Fast food (Y/N)']  # 1 NaN value

# Dropping PCOS (Y/N) column and creating target column y for predictions
X = df.drop(columns=["PCOS (Y/N)"])
y = df["PCOS (Y/N)"]

# Splitting data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.35)

```

```python
# Creating kNN classifier
knn = KNeighborsClassifier(n_neighbors=2)  # Fit the classifier to the data
knn.fit(X_train, y_train)

# Prediction
y_pred=knn.predict(X_test)

# Model Evaluation
print(f"Score in Test Data : {knn.score(X_test,y_test)}")

cm=confusion_matrix(y_test, y_pred)
p_right=cm[0][0]+cm[1][1]
p_wrong=cm[0][1]+cm[1][0]

print(f"Right classification : {p_right}")
print(f"Wrong classification : {p_wrong}")

# Finding kNN accuracy and other measures
print("k Nearest Neighbors model accuracy(in %):", knn.score(X_test, y_test)*100)
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))

# Generating Classification Report for kNN
classi_report = classification_report(y_test, y_pred)
print(classi_report)

# Plotting the confusion matrix for kNN
plt.subplots(figsize=(15, 5))
cf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(cf_matrix/np.sum(cf_matrix), annot=True, annot_kws={'size': 15}, cmap='Pastel2')
plt.title("Confusion Matrix for kNN")
plt.show()

# Plotting ROC curve with AUC
metrics.plot_roc_curve(knn, X_test, y_test)
plt.show()
```

'Classification Report' obtained as code output for the above kNN without feature selection code:

```
Python - kNearestNeighboures.py:34 ✔

k Nearest Neighbors model accuracy(in %): 73.15789473684211
Mean Absolute Error: 0.26842105263157895
Mean Squared Error: 0.26842105263157895
Root Mean Squared Error: 0.5180936716768301
              precision    recall  f1-score   support

           0       0.73      0.96      0.83       130
           1       0.74      0.23      0.35        60

    accuracy                           0.73       190
   macro avg       0.73      0.60      0.59       190
weighted avg       0.73      0.73      0.68       190
```

5.2.3 Performance measures for KNN after feature selection



Accuracy = $\frac{0.18+0.66}{0.18+0.66+0.011+0.14}$ = 0.8459 , i.e. 84.59%

Misclassification error = 1 - 0.8459 = 0.1541 , i.e. 15.41 %

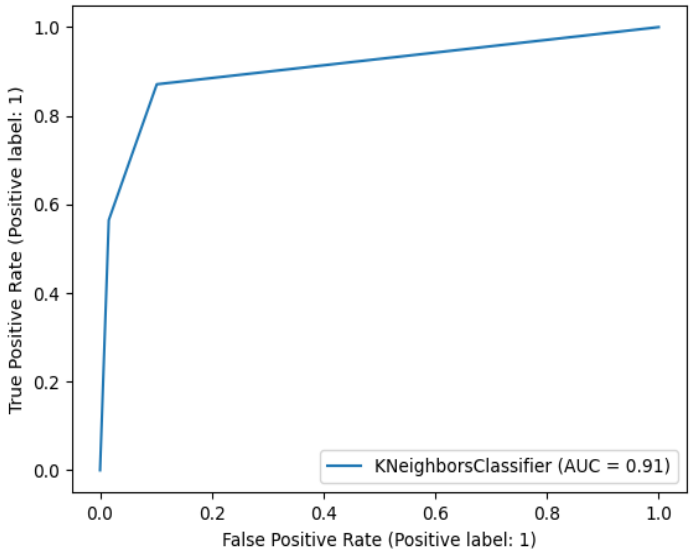
Precision = $\frac{0.18}{0.18+0.011}$ = 0.9424

Recall or tp rate = $\frac{0.18}{0.18+0.14}$ = 0.5625

fp rate = $\frac{0.011}{0.011+0.66}$ = 0.0164

F1 score = $\frac{2*0.011*0.5625}{0.011+0.5625}$ = 0.0216

ROC curve for kNN model with selected 5 features-
As it can be seen from plot, Area under Curve=0.91



The following additional feature selection code with the few changes to previous code was used for the above:

```
# Reading datasets
df_inf = pd.read_csv('C:/Users/nicnnnn/Documents/PCOS_Project/Datasets/PCOS_infertility.csv')
df_noinf = pd.read_csv('C:/Users/nicnnnn/Documents/PCOS_Project/Datasets/PCOS_withoutinfertility.csv')

# Sampling data from df_inf and df_noinf
df_inf.sample(5)
df_noinf.sample(5)

## Feature Selection
# Identifying Features which have more than 0.40 correlation with PCOS(Y/N)
corr_features=df_noinf.corrwith(df_noinf["PCOS (Y/N)"]).abs().sort_values(ascending=False)
# features with correlation more than 0.4
corr_features=corr_features[corr_features>0.4].index
corr_features
df_inf.corrwith(df_inf["PCOS (Y/N)"]).abs()
df_noinf=df_noinf[corr_features]
df_noinf.head()
df_noinf.columns

# Dropping PCOS (Y/N) column and creating target column y for predictions
X = df_noinf.drop(columns=["PCOS (Y/N)"])
y = df_noinf["PCOS (Y/N)"]
```

'Classification Report' obtained as code output for the kNN with feature selection code:

```
Python - kNearestNeighbouresModel_withFeatureSelection.py:16 ✔

Score in Test Data : 0.8473684210526315
Right classification : 161
Wrong classification : 29
k Nearest Neighbors with feature selection model accuracy(in %): 84.73684210526315
Mean Absolute Error: 0.15263157894736842
Mean Squared Error: 0.15263157894736842
Root Mean Squared Error: 0.39068091705043445
              precision    recall  f1-score   support

           0       0.82      0.98      0.90       128
           1       0.95      0.56      0.71        62

    accuracy                           0.85       190
   macro avg       0.88      0.77      0.80       190
weighted avg       0.86      0.85      0.83       190

[Finished in 42.132s]
```

5.3 Decision Tree

5.3.1 Decision tree model building

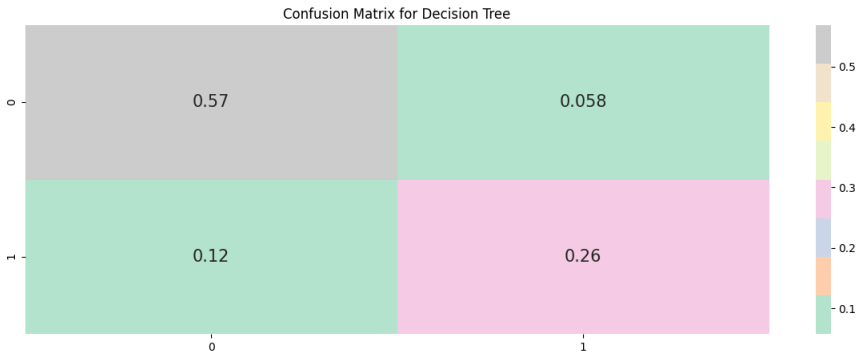
The Decision Tree model with all features considered was built as follows:

9. Splitting training and testing set with test size as 0.35
10. Creating Logistic Regression model and fitting our train and test sets in it
11. Predicting output using test set
12. Model evaluation - finding score in test data, number of right and wrong classifications.
13. Finding accuracy and other measures like Mean Absolute Error, Mean Squared Error, Root Mean Squared Error
14. Classification report generated
15. Confusion matrix plotted
16. ROC curve was plotted and AUC found

The Decision Tree model by selecting features considered was built as follows:

10. Selecting features with correlation >0.40 (5 features- 'Follicle No.(L)', 'Skin darkening', 'Hair growth', 'Weight gain', 'Cycle' satisfied the condition)
11. Splitting training and testing set with test size as 0.35
12. Creating Logistic Regression model and fitting our train and test sets in it
13. Predicting output using test set
14. Model evaluation - finding score in test data, number of right and wrong classifications.
15. Finding accuracy and other measures like Mean Absolute Error, Mean Squared Error, Root Mean Squared Error
16. Classification report generated
17. Confusion matrix plotted
18. ROC curve was plotted and AUC found

5.3.2 Performance measures for Decision Tree with all features considered



$$\text{Accuracy} = \frac{0.26+0.57}{0.26+0.57+0.058+0.12} = 0.8234 \text{ , i.e. } 82.34\%$$

Misclassification error = 1 - 0.8234 = 0.1765 , i.e. 17.65 %

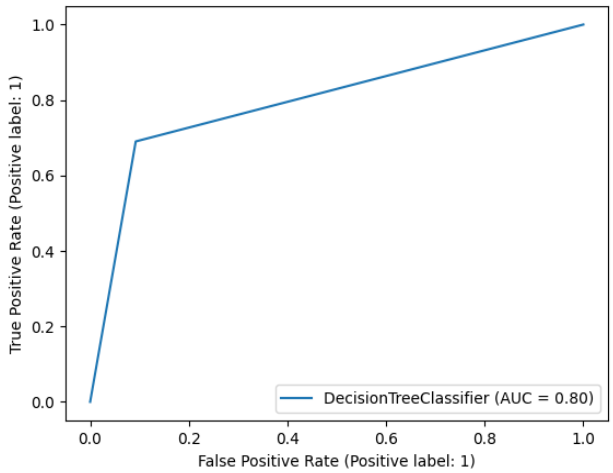
$$\text{Precision} = \frac{0.26}{0.26+0.058} = 0.6842$$

$$\text{Recall or tp rate} = \frac{0.26}{0.26+0.12} = 0.8176$$

$$\text{fp rate} = \frac{0.058}{0.57+0.058} = 0.092$$

$$\text{F1 score} = \frac{2*0.6842*0.8176}{0.6842+0.8176} = 0.7449$$

ROC curve for Decision Tree model with all features considered-
As it can be seen from plot, Area under Curve=0.80



The following code was used for the above:

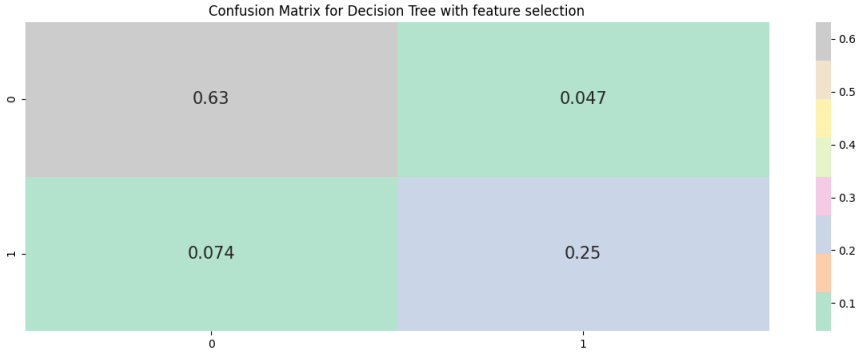
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
import pickle
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

# Reading dataset
df = pd.read_csv('C:/Users/nicnnnn/Documents/PCOS_Project/Datasets/PCOS_withoutinfertility.csv')

# Treating errors in dataset
del df['AMH(ng/mL)']
del df['Unnamed: 42']  # All NaN value
del df['Marraige Status (Yrs)']  # 1 NaN value
del df['Fast food (Y/N)']  # 1 NaN value

# Dropping PCOS (Y/N) column and creating target column y for predictions
X = df.drop(columns=["PCOS (Y/N)"])
y = df["PCOS (Y/N)"]

# Splitting data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.35)

```

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
import pickle
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

# Reading dataset
df = pd.read_csv('C:/Users/nicnnnn/Documents/PCOS_Project/Datasets/PCOS_withoutinfertility.csv')

# Treating errors in dataset
del df['AMH(ng/mL)']
del df['Unnamed: 42']  # All NaN value
del df['Marraige Status (Yrs)']  # 1 NaN value
del df['Fast food (Y/N)']  # 1 NaN value

# Dropping PCOS (Y/N) column and creating target column y for predictions
X = df.drop(columns=["PCOS (Y/N)"])
y = df["PCOS (Y/N)"]

# Splitting data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.35)

```

'Classification Report' obtained as code output for the above Decision Tree without feature selection code:

```
Python - DecisionTree.py:8 ✔

Score in Test Data : 0.8263157894736842
Right classification : 157
Wrong classification : 33
Decision Tree with feature selection accuracy(in %): 82.63157894736842
Mean Absolute Error: 0.1736842105263158
Mean Squared Error: 0.1736842105263158
Root Mean Squared Error: 0.41675437673324534
              precision    recall  f1-score   support

           0       0.83      0.91      0.87       119
           1       0.82      0.69      0.75        71

    accuracy                           0.83       190
   macro avg       0.82      0.80      0.81       190
weighted avg       0.83      0.83      0.82       190

[Finished in 32.393s]
```

5.3.3 Performance measures for Decision Tree after feature selection



Accuracy = $\frac{0.25+0.63}{0.25+0.63+0.047+0.074}$ = 0.8791 , i.e. 87.91%

Misclassification error = 1 - 0.8791 = 0.1208 , i.e. 1.21%

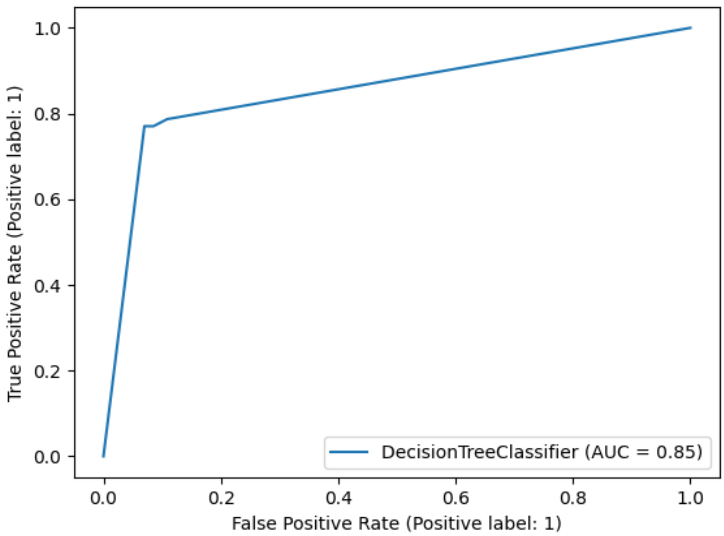
Precision = $\frac{0.25}{0.25+0.047}$ = 0.8417

Recall or tp rate = $\frac{0.25}{0.25+0.074}$ = 0.7716

fp rate = $\frac{0.047}{0.047+0.63}$ = 0.0694

F1 score = $\frac{2*0.8417*0.7716}{0.8417+0.7716}$ = 0.8051

ROC curve for Decision Tree model with selected 5 features-
As it can be seen from plot, Area under Curve=0.85



The following additional feature selection code with the few changes to previous code was used for the above:

```
# Reading datasets
df_inf = pd.read_csv('C:/Users/nicnnnn/Documents/PCOS_Project/Datasets/PCOS_infertility.csv')
df_noinf = pd.read_csv('C:/Users/nicnnnn/Documents/PCOS_Project/Datasets/PCOS_withoutinfertility.csv')

# Sampling data from df_inf and df_noinf
df_inf.sample(5)
df_noinf.sample(5)

## Feature Selection
# Identifying Features which have more than 0.40 correlation with PCOS(Y/N)
corr_features=df_noinf.corrwith(df_noinf["PCOS (Y/N)"]).abs().sort_values(ascending=False)
# features with correlation more than 0.4
corr_features=corr_features[corr_features>0.4].index
corr_features
df_inf.corrwith(df_inf["PCOS (Y/N)"]).abs()
df_noinf=df_noinf[corr_features]
df_noinf.head()
df_noinf.columns

# Dropping PCOS (Y/N) column and creating target column y for predictions
X = df_noinf.drop(columns=["PCOS (Y/N)"])
y = df_noinf["PCOS (Y/N)"]
```

'Classification Report' obtained as code output for the kNN with feature selection code:

```
Python - DecisionTree_withFeatureSelection.py:49 ✔

Score in Test Data : 0.8789473684210526
Right classification : 167
Wrong classification : 23
Decision Tree with feature selection accuracy(in %): 87.89473684210526
Mean Absolute Error: 0.12105263157894737
Mean Squared Error: 0.12105263157894737
Root Mean Squared Error: 0.34792618696922967
              precision    recall  f1-score   support

           0       0.90      0.93      0.91       129
           1       0.84      0.77      0.80        61

    accuracy                           0.88       190
   macro avg       0.87      0.85      0.86       190
weighted avg       0.88      0.88      0.88       190

[Finished in 36.793s]
```

5.4 Ensemble Classifier - Random Forest (Majority vote)

A decision tree is a building block of this random forest model. So we ask a sequence of queries about the available data until one arrives at the decision. In other words a decision tree upon a series of yes/no questions leads to predicted class, since in our case of predicting PCOS we don't have continuous value to be predicted.

Random Forest is a well known machine learning algorithm under the supervised learning technique which can be used for both Classification type of problems and Regression type of problems. In this technique instead of just relying on one decision tree, the algorithm takes predictions from each tree to predict final output (based on majority voting). An important diagnostic measure of this model is to plot the confusion matrix. [8]

5.4.1 Random Forest Model Building

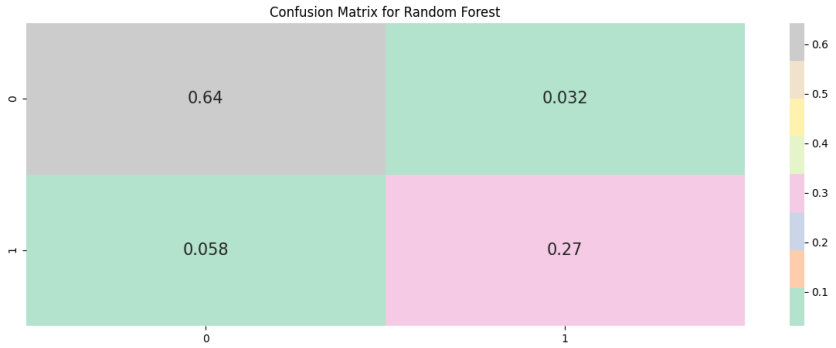
The Random Forest model was built as follows:

1. Splitting training and testing set with test size as 0.35
2. Creating Logistic Regression model and fitting our train and test sets in it
3. Predicting output using test set
4. Model evaluation - finding score in test data, number of right and wrong classifications.
5. Finding accuracy and other measures like Mean Absolute Error, Mean Squared Error, Root Mean Squared Error
6. Classification report generated
7. Confusion matrix plotted
8. ROC curve was plotted and AUC found

The Random Forest model by selecting features considered was built as follows:

1. Selecting features with correlation >0.40 (5 features- 'Follicle No.(L)', 'Skin darkening', 'Hair growth', 'Weight gain', 'Cycle' satisfied the condition)
2. Splitting training and testing set with test size as 0.35
3. Creating Logistic Regression model and fitting our train and test sets in it
4. Predicting output using test set
5. Model evaluation - finding score in test data, number of right and wrong classifications.
6. Finding accuracy and other measures like Mean Absolute Error, Mean Squared Error, Root Mean Squared Error
7. Classification report generated
8. Confusion matrix plotted
9. ROC curve was plotted and AUC found

5.4.2 Performance measures for Random Forest with all features considered



Accuracy = $\frac{0.27+0.64}{0.27+0.64+0.032+0.058}$ = 0.9091 , i.e. 90.91%

Misclassification error = 1 - 0.9091 = 0.0909 , i.e. 9.09%

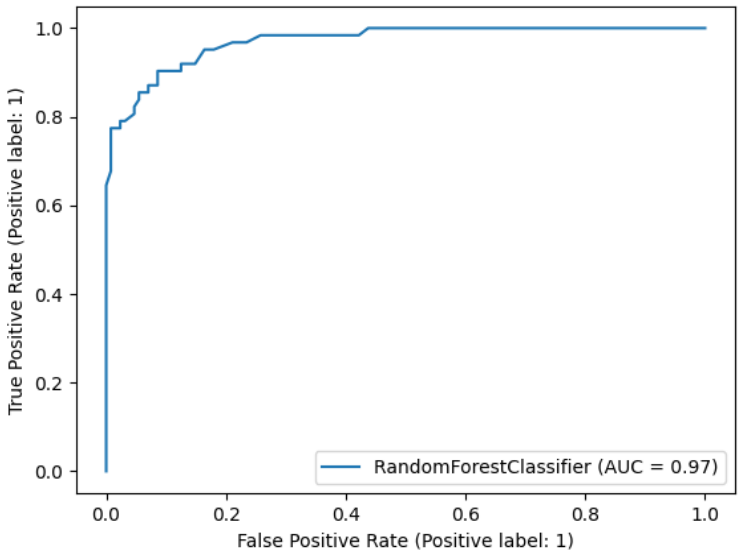
Precision = $\frac{0.27}{0.27+0.032}$ = 0.8940

Recall or tp rate = $\frac{0.27}{0.27+0.058}$ = 0.8231

fp rate = $\frac{0.032}{0.032+0.64}$ = 0.04762

F1 score = $\frac{2*0.8231*0.4762}{0.8231+0.4762}$ = 0.6033

As it can be seen from the ROC curve plot for random forest when all features are considered, Area under Curve=0.97



The following code was used for the above:

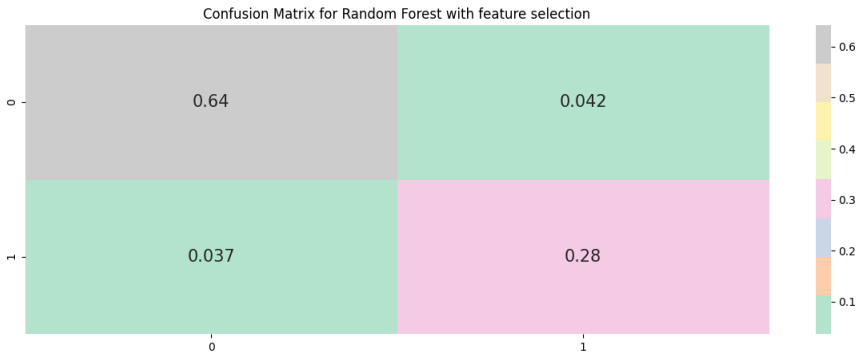
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
import pickle
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

# Reading dataset
df = pd.read_csv('C:/Users/nicnnnn/Documents/PCOS_Project/Datasets/PCOS_withoutinfertility.csv')

# Treating errors in dataset
del df['AMH(ng/mL)']
del df['Unnamed: 42']  # All NaN value
del df['Marraige Status (Yrs)']  # 1 NaN value
del df['Fast food (Y/N)']  # 1 NaN value

# Dropping PCOS (Y/N) column and creating target column y for predictions
X = df.drop(columns=["PCOS (Y/N)"])
y = df["PCOS (Y/N)"]

# Splitting data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.35)
```

```python
# Creating RandomForest Classifier
rfc = RandomForestClassifier()
rfc.fit(X_train, y_train)

# Prediction
y_pred=rfc.predict(X_test)

# Model Evaluation
print(f"Score in Test Data : {rfc.score(X_test,y_test)}")

cm=confusion_matrix(y_test, y_pred)
p_right=cm[0][0]+cm[1][1]
p_wrong=cm[0][1]+cm[1][0]

print(f"Right classification : {p_right}")
print(f"Wrong classification : {p_wrong}")

# Finding Random Forest accuracy and other measures
print("Random Forest model accuracy(in %):", rfc.score(X_test, y_test)*100)
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))

# Generating Classification Report for RandomForestClassifier
classi_report = classification_report(y_test, y_pred)
print(classi_report)

# Plotting the confusion matrix for RandomForestClassifier
plt.subplots(figsize=(15, 5))
cf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(cf_matrix/np.sum(cf_matrix), annot=True, annot_kws={'size': 15}, cmap='Pastel2')
plt.title("Confusion Matrix for Random Forest")
plt.show()

# Plotting ROC curve with AUC
metrics.plot_roc_curve(rfc, X_test, y_test)
plt.show()
```

'Classification Report' obtained as code output for the above Random Forest without feature selection code:

```
Python - RandomForestModel.py:11 ✔

Score in Test Data : 0.9105263157894737
Right classification : 173
Wrong classification : 17
Random Forest model accuracy(in %): 91.05263157894737
Mean Absolute Error: 0.08947368421052632
Mean Squared Error: 0.08947368421052632
Root Mean Squared Error: 0.2991215208080594
              precision    recall  f1-score   support

           0       0.92      0.95      0.93       128
           1       0.89      0.82      0.86        62

    accuracy                           0.91       190
   macro avg       0.91      0.89      0.90       190
weighted avg       0.91      0.91      0.91       190

[Finished in 55.134s]
```

5.4.3 Performance measures for Random Forest with selected features



Accuracy = $\frac{0.28+0.64}{0.28+0.64+0.042+0.037}$ = 0.9206 , i.e. 92.06%

Misclassification error = 1 - 0.9206 = 0.0793 , i.e. 7.93%

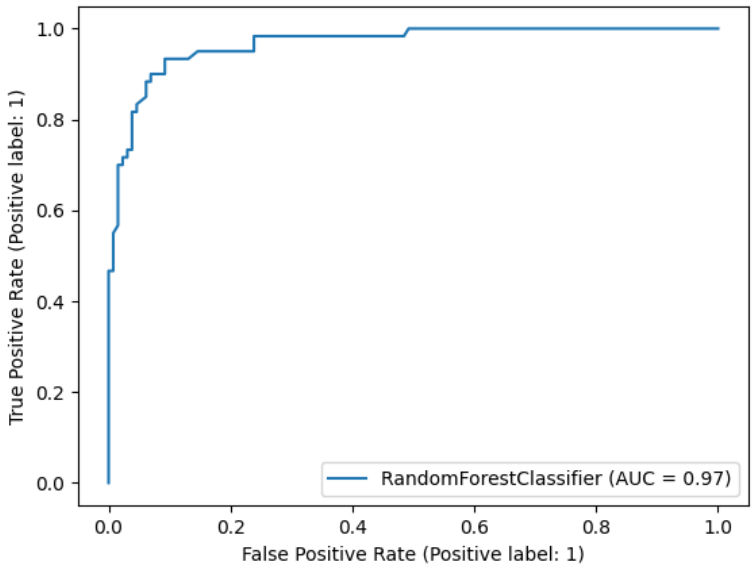
Precision = $\frac{0.28}{0.28+0.042}$ = 0.8695

Recall or tp rate = $\frac{0.28}{0.28+0.037}$ = 0.8832

fp rate = $\frac{0.042}{0.042+0.64}$ = 0.0615

$$\text{F1 score} = \frac{2*0.8695*0.8832}{0.8695+0.8832} = 0.8762$$

ROC curve for Random Forest model with selected 5 features-
As it can be seen from plot, Area under Curve=0.97



The following additional feature selection code with the few changes to previous code was used for the above:

```
# Reading datasets
df_inf = pd.read_csv('C:/Users/nicnnnn/Documents/PCOS_Project/Datasets/PCOS_infertility.csv')
df_noinf = pd.read_csv('C:/Users/nicnnnn/Documents/PCOS_Project/Datasets/PCOS_withoutinfertility.csv')

# Sampling data from df_inf and df_noinf
df_inf.sample(5)
df_noinf.sample(5)

## Feature Selection
# Identifying Features which have more than 0.40 correlation with PCOS(Y/N)
corr_features=df_noinf.corrwith(df_noinf["PCOS (Y/N)"]).abs().sort_values(ascending=False)
# features with correlation more than 0.4
corr_features=corr_features[corr_features>0.4].index
corr_features
df_inf.corrwith(df_inf["PCOS (Y/N)"]).abs()
df_noinf=df_noinf[corr_features]
df_noinf.head()
df_noinf.columns

# Dropping PCOS (Y/N) column and creating target column y for predictions
X = df_noinf.drop(columns=["PCOS (Y/N)"])
y = df_noinf["PCOS (Y/N)"]
```

'Classification Report' obtained as code output for the Random Forest model with feature selection code:

```
Python - RandomForestModel_withFeatureSelection.py:48 ✔

Score in Test Data : 0.9210526315789473
Right classification : 175
Wrong classification : 15
Random Forest with feature selection model accuracy(in %): 92.10526315789474
Mean Absolute Error: 0.07894736842105263
Mean Squared Error: 0.07894736842105263
Root Mean Squared Error: 0.28097574347450816
              precision    recall  f1-score   support

           0       0.95      0.94      0.94       130
           1       0.87      0.88      0.88        60

    accuracy                           0.92       190
   macro avg       0.91      0.91      0.91       190
weighted avg       0.92      0.92      0.92       190

[Finished in 41.146s]
```

5.5 Ensemble Classifier - Adaboost (Boosting algorithm)

An AdaBoost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.

5.5.1 Adaboost Model Building

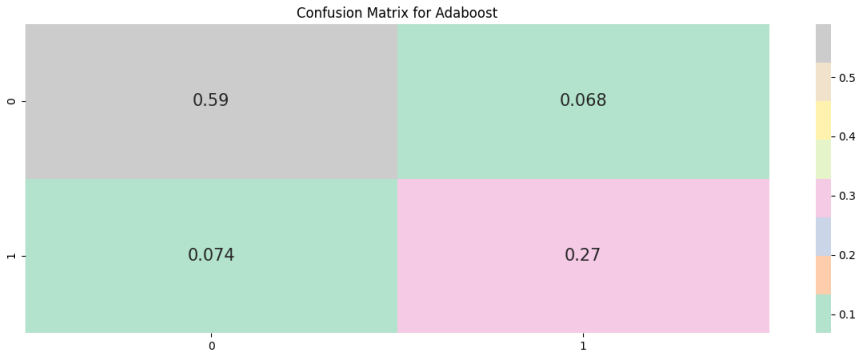
The Adaboost model was built as follows:

9. Splitting training and testing set with test size as 0.35
10. Creating Logistic Regression model and fitting our train and test sets in it
11. Predicting output using test set
12. Model evaluation - finding score in test data, number of right and wrong classifications.
13. Finding accuracy and other measures like Mean Absolute Error, Mean Squared Error, Root Mean Squared Error
14. Classification report generated
15. Confusion matrix plotted
16. ROC curve was plotted and AUC found

The Adaboost model by selecting features considered was built as follows:

10. Selecting features with correlation >0.40 (5 features- 'Follicle No.(L)', 'Skin darkening', 'Hair growth', 'Weight gain', 'Cycle' satisfied the condition)
11. Splitting training and testing set with test size as 0.35
12. Creating Logistic Regression model and fitting our train and test sets in it
13. Predicting output using test set
14. Model evaluation - finding score in test data, number of right and wrong classifications.
15. Finding accuracy and other measures like Mean Absolute Error, Mean Squared Error, Root Mean Squared Error
16. Classification report generated
17. Confusion matrix plotted
18. ROC curve was plotted and AUC found

5.5.2 Performance measures for Adaboost with all features considered



Accuracy = $\frac{0.27+0.59}{0.27+0.59+0.068+0.074}$ = 0.8588 , i.e. 85.88%

Misclassification error = 1 - 0.8588 = 0.1411 , i.e. 14.11%

Precision = $\frac{0.27}{0.27+0.068}$ = 0.7988

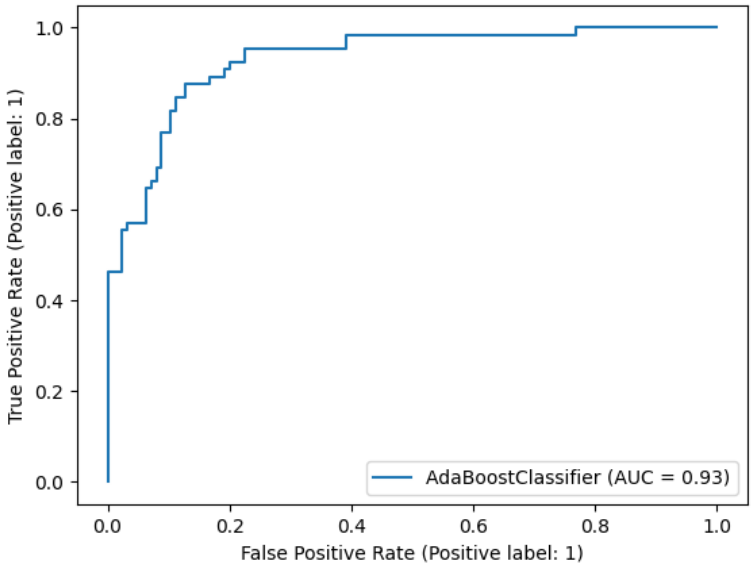
Recall or tp rate = $\frac{0.27}{0.27+0.074}$ = 0.7848

fp rate = $\frac{0.068}{0.068+0.59}$ = 0.1033

F1 score = $\frac{2*0.7988*0.7848}{0.7988+0.7848}$ = 0.7917

ROC curve for Adaboost model with all features considered-

As it can be seen from plot, Area under Curve=0.93



The following code was used for the above:

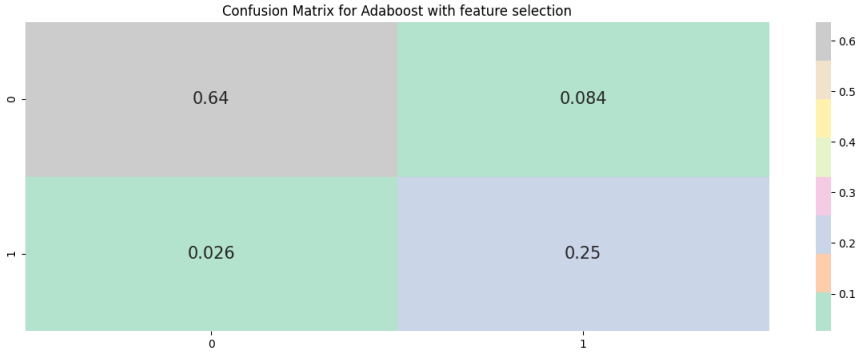
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import AdaBoostClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
import pickle
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

# Reading dataset
df = pd.read_csv('C:/Users/nicnnnn/Documents/PCOS_Project/Datasets/PCOS_withoutinfertility.csv')

# Treating errors in dataset
del df['AMH(ng/mL)']
del df['Unnamed: 42']  # All NaN value
del df['Marraige Status (Yrs)']  # 1 NaN value
del df['Fast food (Y/N)']  # 1 NaN value

# Dropping PCOS (Y/N) column and creating target column y for predictions
X = df.drop(columns=["PCOS (Y/N)"])
y = df["PCOS (Y/N)"]

# Splitting data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.35)

```

```
# Create DecisionTreeClassifier
ada = AdaBoostClassifier()
ada = ada.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = ada.predict(X_test)

# Model Evaluation
print(f"Score in Test Data : {ada.score(X_test,y_test)}")

cm=confusion_matrix(y_test, y_pred)
p_right=cm[0][0]+cm[1][1]
p_wrong=cm[0][1]+cm[1][0]

print(f"Right classification : {p_right}")
print(f"Wrong classification : {p_wrong}")

# LogisticRegression accuracy and other measures
print("Adaboost with feature selection accuracy(in %):", ada.score(X_test, y_test)*100)
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))

# Classification Report for LogisticRegression
classi_report = classification_report(y_test, y_pred)
print(classi_report)

# Plotting the confusion matrix for LogisticRegression
plt.subplots(figsize=(15, 5))
cf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(cf_matrix/np.sum(cf_matrix), annot=True, annot_kws={'size': 15}, cmap='Pastel2')
plt.title("Confusion Matrix for Adaboost")
plt.show()

# Plotting ROC curve with AUC
metrics.plot_roc_curve(ada, X_test, y_test)
plt.show()
```

'Classification Report' obtained as code output for the above Random Forest without feature selection code:

```
Python - Adaboost.py:57 ✔

Score in Test Data : 0.8578947368421053
Right classification : 163
Wrong classification : 27
Adaboost with feature selection accuracy(in %): 85.78947368421052
Mean Absolute Error: 0.14210526315789473
Mean Squared Error: 0.14210526315789473
Root Mean Squared Error: 0.37696851746252596
              precision    recall  f1-score   support

           0       0.89      0.90      0.89       125
           1       0.80      0.78      0.79        65

    accuracy                           0.86       190
   macro avg       0.84      0.84      0.84       190
weighted avg       0.86      0.86      0.86       190

[Finished in 23.864s]
```

5.5.3 Performance measures for Adaboost with selected features



Accuracy = $\frac{0.25+0.64}{0.25+0.64+0.084+0.026}$ = 0.89 , i.e. 89%

Misclassification error = 1 - 0.89 = 0.11 , i.e. 11%

Precision = $\frac{0.25}{0.25+0.084}$ = 0.7485

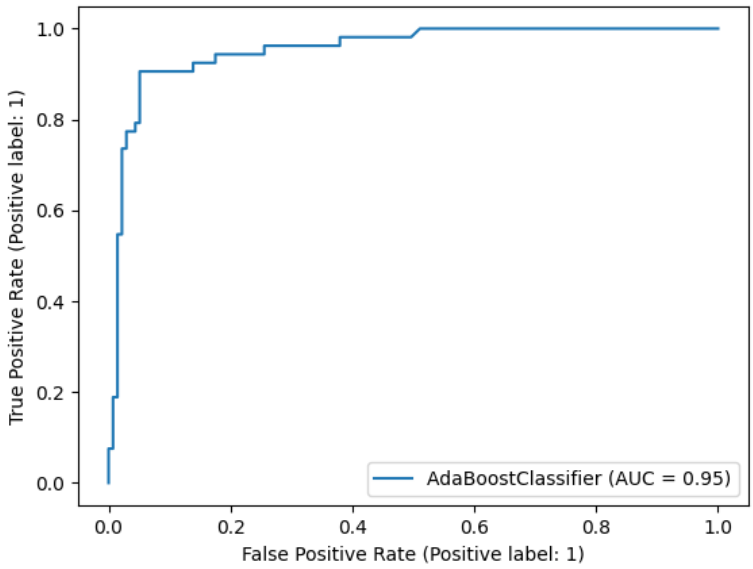
Recall or tp rate = $\frac{0.25}{0.25+0.026}$ = 0.91

fp rate = $\frac{0.084}{0.084+0.64}$ = 0.1160

F1 score = $\frac{2*0.91*0.7485}{0.7485+0.91}$ = 0.82

ROC curve for Adaboost with selected 5 features-

As it can be seen from plot, Area under Curve=0.95



The following additional feature selection code with the few changes to previous code was used for the above:

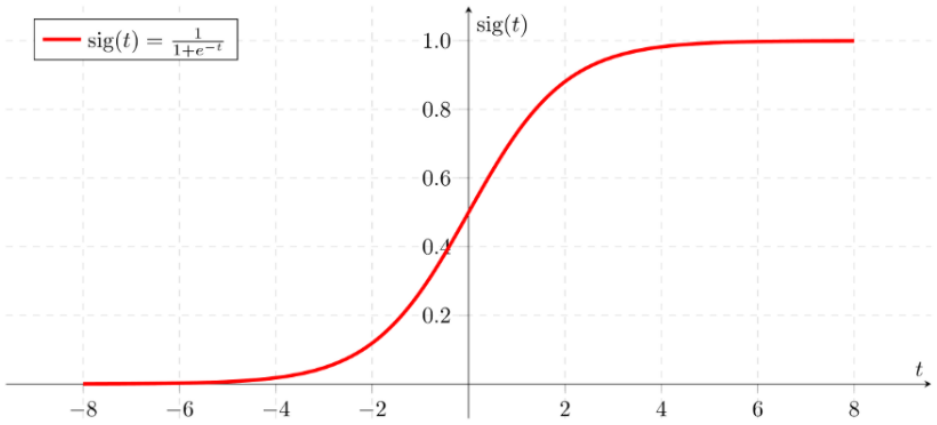
```
# Reading datasets
df_inf = pd.read_csv('C:/Users/nicnnnn/Documents/PCOS_Project/Datasets/PCOS_infertility.csv')
df_noinf = pd.read_csv('C:/Users/nicnnnn/Documents/PCOS_Project/Datasets/PCOS_withoutinfertility.csv')

# Sampling data from df_inf and df_noinf
df_inf.sample(5)
df_noinf.sample(5)

## Feature Selection
# Identifying Features which have more than 0.40 correlation with PCOS(Y/N)
corr_features=df_noinf.corrwith(df_noinf["PCOS (Y/N)"]).abs().sort_values(ascending=False)
# features with correlation more than 0.4
corr_features=corr_features[corr_features>0.4].index
corr_features
df_inf.corrwith(df_inf["PCOS (Y/N)"]).abs()
df_noinf=df_noinf[corr_features]
df_noinf.head()
df_noinf.columns

# Dropping PCOS (Y/N) column and creating target column y for predictions
X = df_noinf.drop(columns=["PCOS (Y/N)"])
y = df_noinf["PCOS (Y/N)"]
```

'Classification Report' obtained as code output for the Random Forest model with feature selection code:

```
Python - Adaboost_withFeatureSelection.py:73 ✔

Score in Test Data : 0.8894736842105263
Right classification : 169
Wrong classification : 21
Adaboost with feature selection accuracy(in %): 88.94736842105263
Mean Absolute Error: 0.11052631578947368
Mean Squared Error: 0.11052631578947368
Root Mean Squared Error: 0.33245498310218435
              precision    recall  f1-score   support

           0       0.96      0.88      0.92       137
           1       0.75      0.91      0.82        53

    accuracy                           0.89       190
   macro avg       0.86      0.89      0.87       190
weighted avg       0.90      0.89      0.89       190

[Finished in 30.011s]
```

5.6 Logistic Regression

Logistic regression is in effect a way found by statisticians to adopt regression models in modeling a binary outcome like yes or no, which may not necessarily seem like a regression task.



Here the hypothesis proposed is a sigmoid function as shown above, the output from the hypothesis is the estimated probability.

Output = 0 or 1
Hypothesis => Z = Wx + B
h(x) = sigmoid (Z)

Therefore, if Z goes to infinity PCOS is predicted (1) on the contrary if Z goes to negative infinity the person is predicted to not have PCOS (0). The logistic regression model not just is a classification model but also provides the physician or specialist with probabilities, this is a big advantage over previous models which can only provide the final classification. So, in conclusion, this technique can be restated as 'what is the probability that an instance belongs to class q?' [7]

5.6.1 Justification for not using Linear Regression

Logistic Regression is used when the target dependent variable is categorical, in our case, it is 'Yes or 1' (the person has PCOS) and 'No or 0' (the person does not have PCOS) which makes it in effect a binary logistic regression. In such cases like ours linear regression is not employed, because doing so would warrant setting up a threshold based on which classification is done. In our case the class is sensitive, since it is predicting a disease, so predicted continuous values like 0.4 or threshold value of 0.5 would mean a prediction of the person not having PCOS. This would have serious consequences in real

time. Therefore, a linear regression is not suitable for classification problems as it yields predictions that are not necessarily restricted between 0 and 1 which then brings logistic regression into picture where values strictly range from 0 to 1. [9]

5.6.2 Logistic Regression Model Building

The coefficient in a logistic regression algorithm is estimated from our training data set (35% in our case) which is done using maximum-likelihood estimation. In statistics, MLE is a parameter estimation method of a probability distribution so under the assumed statistical model the observed data is most probable. Therefore, data preparation and distribution becomes crucial in a Logistic regression algorithm. Hence, some important assumptions must be made in this algorithm.

Data preparation for Logistic Regression:

- **Binary output variable** - In our case we have it already snapped into a 1 (patient has PCOS) and 0 (patient does not have PCOS) classification.
- **Noise Removal** - In section 3.2.6 outliers and such instances have been removed data so only meaningful variables are included.
- **Removal of correlated input** - In section 3.2 we illustrated the process of "feature extraction" and "feature selection" which becomes extremely crucial in Logistic Regression. The code used in this section,

  1. Calculated the pairwise correlations between all inputs and between inputs and output.
  2. Identifies all features which have more than 0.40 correlation with PCOS(Y/N).
  3. Those identified features are then extracted and we proceed to apply the Logistic Regression algorithm.

The previous section 4.8.3 has the observation for this algorithm with all features and immediately following that in this current section 4.8.4 is the observation of this algorithm after feature selection and extraction. The main logic behind doing so is the possibility of the model to "overfit" if we have multiple highly-correlated inputs. This may lead to a failure to converge hence, feature extraction played a very important role in this algorithm.

The Logistic Regression model with all features considered was built as follows:

1. Splitting training and testing set with test size as 0.35
2. Creating Logistic Regression model and fitting our train and test sets in it
3. Predicting output using test set

4. Model evaluation - finding score in test data, number of right and wrong classifications.
5. Finding accuracy and other measures like Mean Absolute Error, Mean Squared Error, Root Mean Squared Error
6. Classification report generated
7. Confusion matrix plotted
8. ROC curve was plotted and AUC found

The Logistic Regression model by selecting features considered was built as follows:

1. Selecting features with correlation >0.40 (5 features- 'Follicle No.(L)', 'Skin darkening', 'Hair growth', 'Weight gain', 'Cycle' satisfied the condition)
2. Splitting training and testing set with test size as 0.35
3. Creating Logistic Regression model and fitting our train and test sets in it
4. Predicting output using test set
5. Model evaluation - finding score in test data, number of right and wrong classifications.
6. Finding accuracy and other measures like Mean Absolute Error, Mean Squared Error, Root Mean Squared Error
7. Classification report generated
8. Confusion matrix plotted
9. ROC curve was plotted and AUC found

5.6.3 Performance measures for Logistic Regression with all features considered

Accuracy = $\frac{0.25+0.63}{0.25+0.63+0.037+0.079}$ = 0.8631 , i.e. 86.31%

Misclassification error = 1 - 0.8631 = 0.1366 , i.e. 13.69%

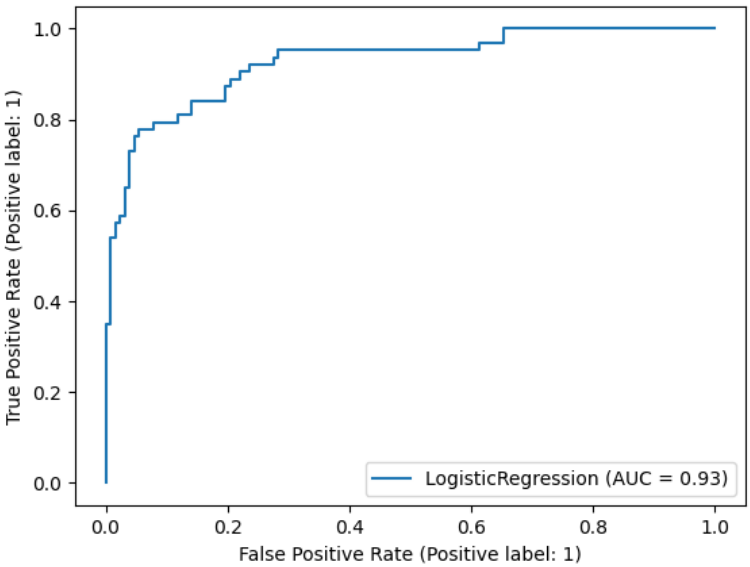
Precision = $\frac{0.25}{0.25+0.037}$ = 0.81

Recall or tp rate = $\frac{0.25}{0.25+0.079}$ = 0.73

fp rate = $\frac{0.037}{0.037+0.63}$ = 0.0554

F1 score = $\frac{2*0.81*0.73}{0.73+0.81}$ = 0.77

ROC curve for Logistic Regression model with all features considered-
As it can be seen from plot, Area under Curve=0.93



The following code was used for the above:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
import pickle
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

# Reading dataset
df = pd.read_csv('C:/Users/nicnnnn/Documents/PCOS_Project/Datasets/PCOS_withoutinfertility.csv')

# Treating errors in dataset
del df['AMH(ng/mL)']
del df['Unnamed: 42']  # All NaN value
del df['Marraige Status (Yrs)']  # 1 NaN value
del df['Fast food (Y/N)']  # 1 NaN value

# Dropping PCOS (Y/N) column and creating target column y for predictions
X = df.drop(columns=["PCOS (Y/N)"])
y = df["PCOS (Y/N)"]

# Splitting data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.35)

```

```python
# Create LogisticRegression classifier
lrm=LogisticRegression()
lrm.fit(X_train,y_train)

# Prediction
y_pred=lrm.predict(X_test)

# Model Evaluation
print(f"Score in Test Data : {lrm.score(X_test,y_test)}")

cm=confusion_matrix(y_test, y_pred)
p_right=cm[0][0]+cm[1][1]
p_wrong=cm[0][1]+cm[1][0]

print(f"Right classification : {p_right}")
print(f"Wrong classification : {p_wrong}")

# LogisticRegression accuracy and other measures
print("Logistic Regression with feature selection accuracy(in %):", lrm.score(X_test, y_test)*100)

# Classification Report for LogisticRegression
classi_report = classification_report(y_test, y_pred)
print(classi_report)

# Plotting the confusion matrix for LogisticRegression
plt.subplots(figsize=(15, 5))
cf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(cf_matrix/np.sum(cf_matrix), annot=True, annot_kws={'size': 15}, cmap='Pastel2')
plt.title("Confusion Matrix for LogisticRegression")
plt.show()

# Plotting ROC curve with AUC
metrics.plot_roc_curve(lrm, X_test, y_test)
plt.show()
```

'Classification Report' obtained as code output for the above Logistic Regression without feature selection code:

```
Python - LogisticRegression.py:52 ✔
Score in Test Data : 0.8631578947368421
Right classification : 164
Wrong classification : 26
Logistic Regression with feature selection accuracy(in %): 86.31578947368422
              precision    recall  f1-score   support

           0       0.88      0.92      0.90       130
           1       0.81      0.73      0.77        60

    accuracy                           0.86       190
   macro avg       0.85      0.83      0.84       190
weighted avg       0.86      0.86      0.86       190

[Finished in 54.16s]
```

5.6.4 Performance measures for Logistic Regression after feature selection

Accuracy = $\frac{0.34+0.56}{0.34+0.56+0.037+0.063}$ = 0.9 , i.e. 90.00 %

Misclassification error = 1 - 0.90 = 0.10 , i.e. 10.00%

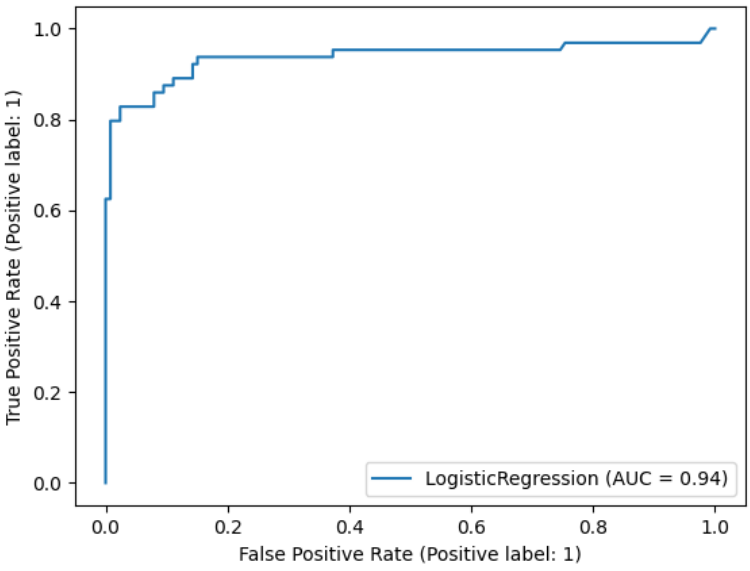
Precision = $\frac{0.34}{0.34+0.037}$ = 0.9018

Recall or tp rate = $\frac{0.34}{0.34+0.063}$ = 0.8436

fp rate = $\frac{0.037}{0.037+0.56}$ = 0.0619

F1 score = $\frac{2*0.9018*0.8436}{0.8436+0.9018}$ = 0.8717

ROC curve for Logistic Regression model with selected 5 features-
As it can be seen from plot, Area under Curve=0.94



The following additional feature selection code with the few changes to previous code was used for the above:

```
# Reading datasets
df_inf = pd.read_csv('C:/Users/nicnnnn/Documents/PCOS_Project/Datasets/PCOS_infertility.csv')
df_noinf = pd.read_csv('C:/Users/nicnnnn/Documents/PCOS_Project/Datasets/PCOS_withoutinfertility.csv')

# Sampling data from df_inf and df_noinf
df_inf.sample(5)
df_noinf.sample(5)

## Feature Selection
# Identifying Features which have more than 0.40 correlation with PCOS(Y/N)
corr_features=df_noinf.corrwith(df_noinf["PCOS (Y/N)"]).abs().sort_values(ascending=False)
# features with correlation more than 0.4
corr_features=corr_features[corr_features>0.4].index
corr_features
df_inf.corrwith(df_inf["PCOS (Y/N)"]).abs()
df_noinf=df_noinf[corr_features]
df_noinf.head()
df_noinf.columns

# Dropping PCOS (Y/N) column and creating target column y for predictions
X = df_noinf.drop(columns=["PCOS (Y/N)"])
y = df_noinf["PCOS (Y/N)"]
```

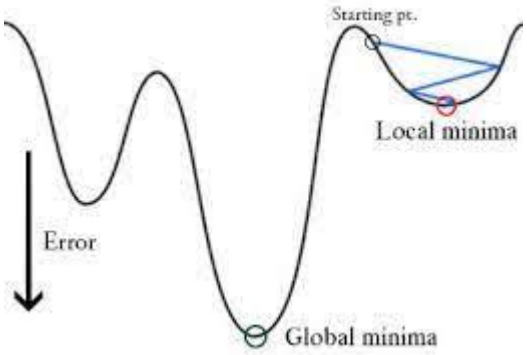
'Classification Report' obtained as code output for the Logistic Regression model with feature selection code:

```
Python - LogisticRegressionModel_withFeatureSelection.py:59 ✔

Score in Test Data : 0.9
Right classification : 171
Wrong classification : 19
Logistic Regression with feature selection accuracy(in %): 90.0
Mean Absolute Error: 0.1
Mean Squared Error: 0.1
Root Mean Squared Error: 0.31622776601683794
              precision    recall  f1-score   support

           0       0.91      0.94      0.93       126
           1       0.87      0.83      0.85        64

    accuracy                           0.90       190
   macro avg       0.89      0.88      0.89       190
weighted avg       0.90      0.90      0.90       190

[Finished in 35.604s]
```

5.6.5 Log-loss or Binary Cross Entropy

We don't use Mean Squared Error' as a cost function in Logistic Regression. But the question arises, why?

In Logistic Regression, Ŷi is a nonlinear function($\hat{Y}$=1/1+ e-z), if we put this in the above MSE equation it will give a non-convex function as shown:



- When we try to optimize values using gradient descent it will create complications to find global minima.
- Another reason is in classification problems, we have target values like 0/1, So ($\hat{Y}$-*Y)2* will always be in between 0-1 which can make it very difficult to keep track of the errors and it is difficult to store high precision floating numbers.

The cost function used in Logistic Regression is Log Loss.

Log Loss is the negative average of the log of corrected predicted probabilities for each instance. It is the most important classification metric based on probabilities. It's hard to interpret raw log-loss values, but log-loss is still a good metric for comparing models. For any given problem, a lower log loss value means better predictions.

- **Log Loss function**

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

where y is the label (1 for PCOS-yes and 0 for PCOS-no) and p(y) is the predicted probability of the point being PCOS-yes for all N points.

The formula tells us that for each PCOS-yes point, it adds log(p(y)) to the loss, that is, the log probability of it being green. Conversely, it adds *log(1-p(y))*, that is, the log probability of it being PCOS-no.

Besides, what does entropy have to do with all this? Why are we taking log of probabilities in the first place? Are some questions we answer further.

- Entropy

Entropy is the measure of uncertainty associated with a given distribution.

If we knew exactly half of the points were PCOS-yes and the other half were PCOS-no, it would be the worst-case scenario. We would have absolutely no edge in guessing whether PCOS-yes or PCOS-no. For this case, entropy is given by:

$$H(q) = log(2)$$

For every other case in between, we can compute the entropy of a distribution, like our q(y), using the formula below, where C is the number of classes:

$$H(q) = -\sum_{c=1}^{C} q(y_c) \cdot log(q(y_c))$$

So, if we *know* the true distribution of a random variable, we can compute its entropy.

- Cross-Entropy

Let's assume our points follow this other distribution p(y). But we know they are actually coming from the true (*unknown*) distribution q(y), right? If we compute entropy like this, we are actually computing the cross-entropy between both distributions:

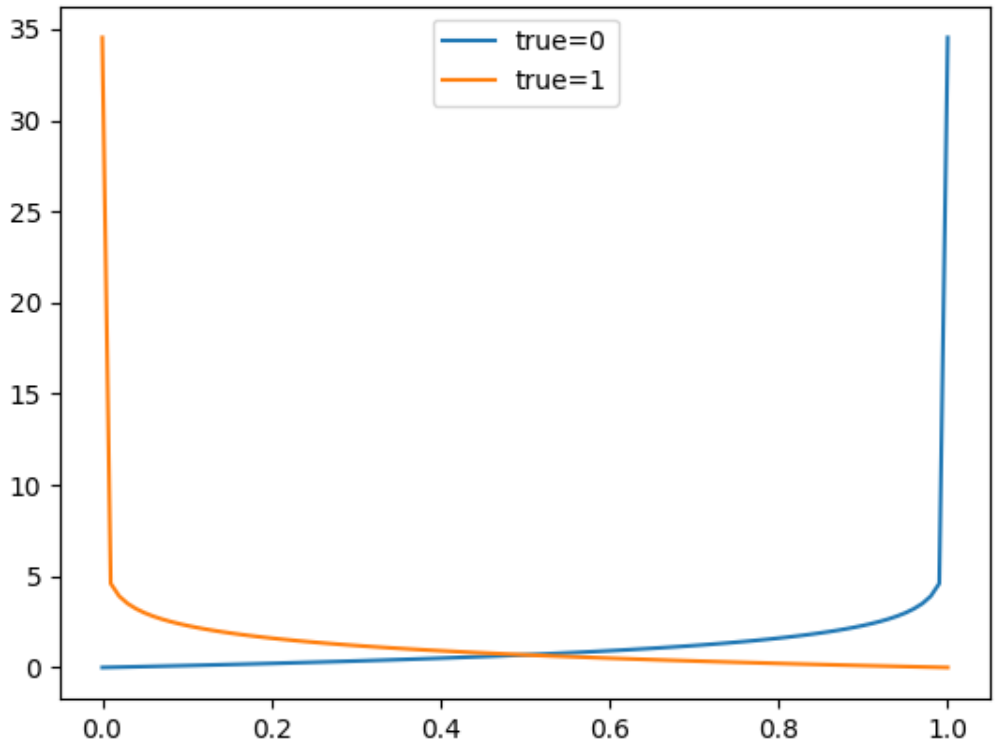
$$H_p(q) = -\sum_{c=1}^{C} q(y_c) \cdot log(p(y_c))$$

If we, somewhat miraculously, *match* p(y) *to* q(y) *perfectly*, the computed values for both cross-entropy *an*d entropy will match as well. Since this is likely never happening, cross-entropy will have a BIGGER value than the entropy computed on the true distribution.

Cross-entropy minus Entropy is:

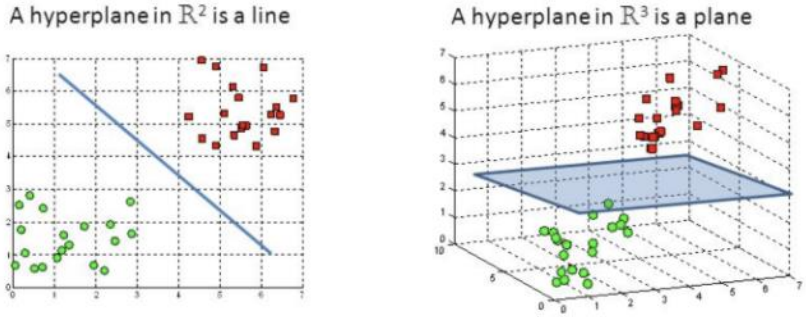
$$H_p(q) - H(q) >= 0$$

Line plot of evaluating predictions with log loss:

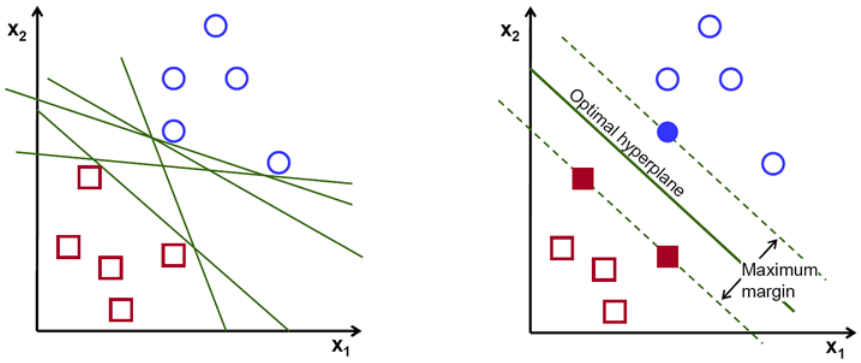


5.7 Support Vector Machine

The basic concept of SVM learning technique can be summarised as the technique to find a good learning boundary while maximizing the margin which is the distance between the closest samples that correspond to different classes. SVM is a supervised machine learning algorithm which was originally designed for binary classification, therefore, it would only be relevant to implement this technique for this project. In this algorithm, each data item is plotted as a point in the n-dimensional space (where n is the number of features) and each coordinate value corresponds to the value of each feature. Therefore, we could say the term “Support Vectors" are nothing but the coordinates of individual observation. Hence, SVM is a classifier mechanism that can segregate two classes in an efficient manner and is generally found to produce significant accuracy with less computation power.



The challenge in this technique lies in identifying the most appropriate hyper-plane that separates the two classes better, the hyper-plane could be linear or even non-linear. “Hyperplanes" are decision boundaries that helped classify the data points. The SVM “kernels” become very useful in non-linear separation problems where these functions transform low dimensional input space into high dimensional space.



To separate two classes of data points there are several hyperplanes which could be chosen. The aim however is to find the plane that has the maximum distance between

data points of both the classes. Maximizing the margin distance strengthens the future data classification likelihood. It is also important to note that the dimensions of the hyperplane depend on the number of features. The above few illustrations are just for understanding. A hyperplane is a simple line with only 2 features and enters 3 dimensional space for 3 features, however in our case for predicting PCOS the number of features increases and becomes difficult to imagine.

In section 4. 8 We saw in detail how logistic regression uses the sigmoid function to squash the value between 0 to 1 range. So a threshold value decides what label we assign. However in SVM, the reinforcement range of value is between -1 and 1 range which acts as the margin.

5.7.1 Support Vector Machine Model Building

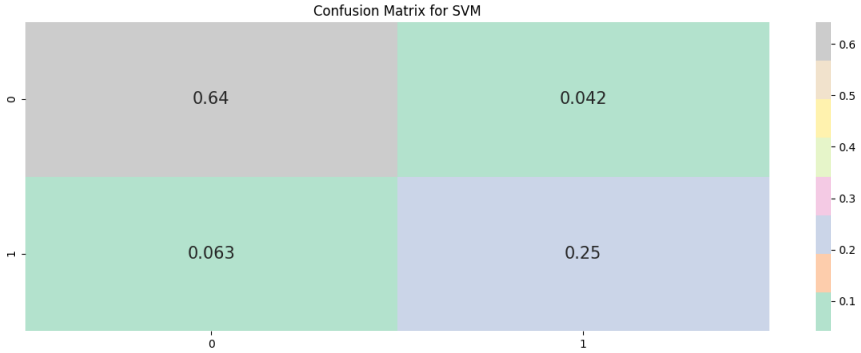
The SVM model with all features considered was built as follows:

9. Splitting training and testing set with test size as 0.35
10. Creating Logistic Regression model and fitting our train and test sets in it
11. Predicting output using test set
12. Model evaluation - finding score in test data, number of right and wrong classifications.
13. Finding accuracy and other measures like Mean Absolute Error, Mean Squared Error, Root Mean Squared Error
14. Classification report generated
15. Confusion matrix plotted
16. ROC curve was plotted and AUC found

The SVM model by selecting features considered was built as follows:

10. Selecting features with correlation >0.40 (5 features- 'Follicle No.(L)', 'Skin darkening', 'Hair growth', 'Weight gain', 'Cycle' satisfied the condition)
11. Splitting training and testing set with test size as 0.35
12. Creating Logistic Regression model and fitting our train and test sets in it
13. Predicting output using test set
14. Model evaluation - finding score in test data, number of right and wrong classifications.
15. Finding accuracy and other measures like Mean Absolute Error, Mean Squared Error, Root Mean Squared Error
16. Classification report generated
17. Confusion matrix plotted
18. ROC curve was plotted and AUC found

5.7.2 Performance measures for SVM with all features considered



$$\text{Accuracy} = \frac{0.25+0.64}{0.25+0.64+0.042+0.063} = 0.8789 \text{ , i.e. } 89.32\%$$

Misclassification error = 1 - 0.8789 = 0.1067 , i.e. 10.67%

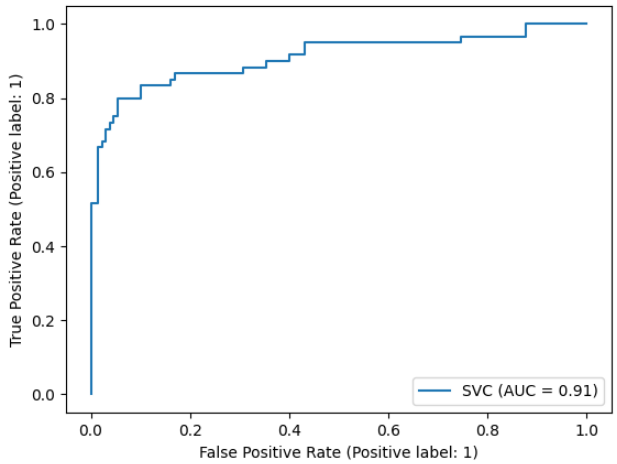
$$\text{Precision} = \frac{0.25}{0.25+0.042} = 0.86$$

$$\text{Recall or tp rate} = \frac{0.25}{0.25+0.084} = 0.80$$

$$\text{fp rate} = \frac{0.042}{0.042+0.64} = 0.0654$$

$$\text{F1 score} = \frac{2*0.86*0.80}{0.86+0.80} = 0.83$$

ROC curve for SVM model with all features considered-
As it can be seen from plot, Area under Curve=0.91



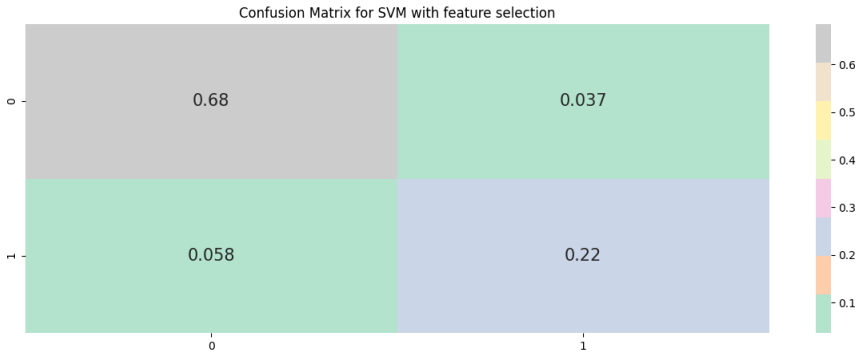
The following code was used for the above:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn import metrics
import pickle
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

# Reading dataset
df = pd.read_csv('C:/Users/nicnnnn/Documents/PCOS_Project/Datasets/PCOS_withoutinfertility.csv')

# Treating errors in dataset
del df['AMH(ng/mL)']
del df['Unnamed: 42']  # All NaN value
del df['Marraige Status (Yrs)']  # 1 NaN value
del df['Fast food (Y/N)']  # 1 NaN value

# Dropping PCOS (Y/N) column and creating target column y for predictions
X = df.drop(columns=["PCOS (Y/N)"])
y = df["PCOS (Y/N)"]

# Splitting data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.35)

# Create SVM classifier
svc = svm.SVC(kernel='linear')
svc.fit(X_train, y_train)

# Prediction
y_pred = svc.predict(X_test)

# Model Evaluation
print(f"Score in Test Data : {svc.score(X_test,y_test)}")

cm=confusion_matrix(y_test, y_pred)
p_right=cm[0][0]+cm[1][1]
p_wrong=cm[0][1]+cm[1][0]

print(f"Right classification : {p_right}")
print(f"Wrong classification : {p_wrong}")

# RandomForestClassifier accuracy and other measures
print("SVM model accuracy(in %):", svc.score(X_test, y_test)*100)
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))

# Classification Report for RandomForestClassifier
classi_report = classification_report(y_test, y_pred)
print(classi_report)

# Plotting the confusion matrix for RandomForestClassifier
plt.subplots(figsize=(15, 5))
cf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(cf_matrix/np.sum(cf_matrix), annot=True, annot_kws={'size': 15}, cmap='Pastel2')
plt.title("Confusion Matrix for SVM")
plt.show()

# ROC curve with svm
metrics.plot_roc_curve(svc, X_test, y_test)
plt.show()
```

'Classification Report' obtained as code output for the above Logistic Regression without feature selection code:

```
Python - SVMModel.py:29 ✔

Score in Test Data : 0.8947368421052632
Right classification : 170
Wrong classification : 20
SVM model accuracy(in %): 89.47368421052632
Mean Absolute Error: 0.10526315789473684
Mean Squared Error: 0.10526315789473684
Root Mean Squared Error: 0.3244428422615251
              precision    recall  f1-score   support

           0       0.91      0.94      0.92       130
           1       0.86      0.80      0.83        60

    accuracy                           0.89       190
   macro avg       0.88      0.87      0.88       190
weighted avg       0.89      0.89      0.89       190

[Finished in 273.524s]
```

5.7.3 SVM performance measures after feature selection



Accuracy = $\frac{0.22+0.68}{0.22+0.68+0.037+0.058}$ = 0.9034 , i.e. 90.34%

Misclassification error = 1 - 0.9034 = 0.0965 , i.e. 9.65%

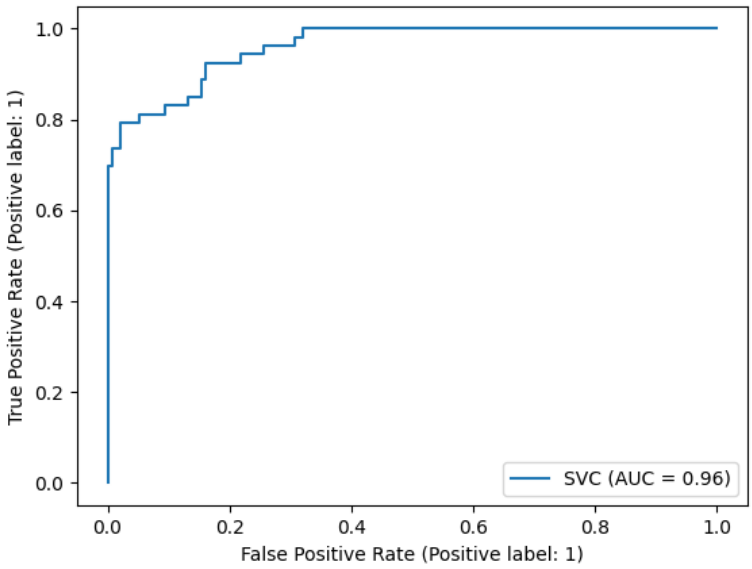
Precision = $\frac{0.22}{0.22+0.037}$ = 0.86

Recall or tp rate = $\frac{0.22}{0.22+0.058}$ = 0.79

fp rate = $\frac{0.037}{0.037+0.68}$ = 0.0516

F1 score = $\frac{2*0.86*0.79}{0.79+0.86}$ = 0.82

ROC curve for SVM model with selected 5 features-
As it can be seen from plot, Area under Curve=0.96



The following is the feature selection and extraction code calculating performance measure outputs for the above confusion matrix for SVM. The following additional feature selection code with the few changes to previous code was used for the above:

```
# Reading datasets
df_inf = pd.read_csv('C:/Users/nicnnnn/Documents/PCOS_Project/Datasets/PCOS_infertility.csv')
df_noinf = pd.read_csv('C:/Users/nicnnnn/Documents/PCOS_Project/Datasets/PCOS_withoutinfertility.csv')

# Sampling data from df_inf and df_noinf
df_inf.sample(5)
df_noinf.sample(5)

## Feature Selection
# Identifying Features which have more than 0.40 correlation with PCOS(Y/N)
corr_features=df_noinf.corrwith(df_noinf["PCOS (Y/N)"]).abs().sort_values(ascending=False)
# features with correlation more than 0.4
corr_features=corr_features[corr_features>0.4].index
corr_features
df_inf.corrwith(df_inf["PCOS (Y/N)"]).abs()
df_noinf=df_noinf[corr_features]
df_noinf.head()
df_noinf.columns

# Dropping PCOS (Y/N) column and creating target column y for predictions
X = df_noinf.drop(columns=["PCOS (Y/N)"])
y = df_noinf["PCOS (Y/N)"]
```

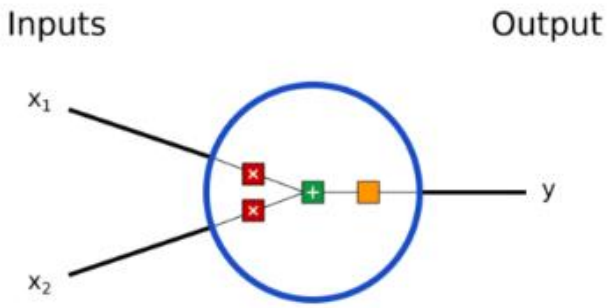
‘Classification Report’ obtained as code output for the SVM model with feature selection code:

Python - SVMModel_withFeatureSelection.py:6 ✔

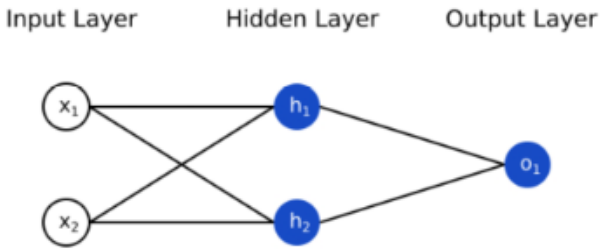
```
Score in Test Data : 0.9052631578947369
Right classification : 172
Wrong classification : 18
SVM with feature selection model accuracy(in %): 90.52631578947368
Mean Absolute Error: 0.09473684210526316
Mean Squared Error: 0.09473684210526316
Root Mean Squared Error: 0.30779350562554625
              precision    recall  f1-score   support

           0       0.92      0.95      0.94       137
           1       0.86      0.79      0.82        53

    accuracy                           0.91       190
   macro avg       0.89      0.87      0.88       190
weighted avg       0.90      0.91      0.90       190

[Finished in 50.437s]
```

5.8 Experimenting with Neural Network

Neural networks are yet another set of algorithms that has been inspired from nature, this time from the biological neural network of a human body. They employ a style of parallel computation inspired by neurons and their adaptive connections. Neural networks are themselves general function approximations, which we attempt to apply to our problem of PCOS prediction. Therefore, neurons are the basic building blocks and form the basic unit of a neural network. Consider neuron as a 'bloackbox' which takes input, does some math and produces one output. A 2-input neuron is illustrated below



Neural network is many such neurons connected together. Therefore, a simple neural network is illustrated below which has 2 inputs, a hidden layer with 2 neurons (h1 and h2) and an output layer with 1 neuron. There can be multiple hidden layers between the first and last (output) layer. A neural network may contain any number of neurons in those layers



These networks have weights and our goal when training a model is to minimize the loss of a neural network. We do so by altering the network's weights and biases to influence the prediction. Here, just like logistic regression, the sigmoid function is used most commonly to turn the unbounded input into an output with a predictable form.

5.8.1 Multi-Layer Perceptron Transfer

This model optimizes the log-loss function using LBFGS or stochastic gradient descent. Multi-layer Perceptron (MLP) is a supervised learning algorithm that learns a function $f(\cdot):Rm \rightarrow Ro$ by training on a dataset, where m is the number of dimensions for input and o is the number of dimensions for output. Given a set of features X=x1,x2,...,xm and a target y, it can learn a nonlinear function approximator for either classification or regression. It is different from logistic regression, in that between the input and the output layer, there can be one or more non-linear layers, called hidden layers. Figure 1 shows a one hidden layer MLP with scalar output.

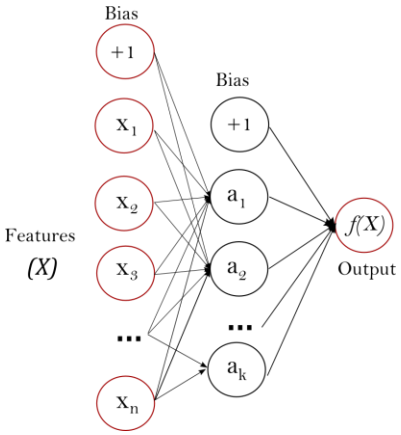


Fig: One hidden layer MLP.

The leftmost layer, known as the input layer, consists of a set of neurons {xi|x1,x2,...,xm} representing the input features. Each neuron in the hidden layer transforms the values from the previous layer with a weighted linear summation W1x1+w2x2+...+wmxm, followed by a non-linear activation function $g(\cdot):R \rightarrow R$ - like the hyperbolic tan function. The output layer receives the values from the last hidden layer and transforms them into output values.

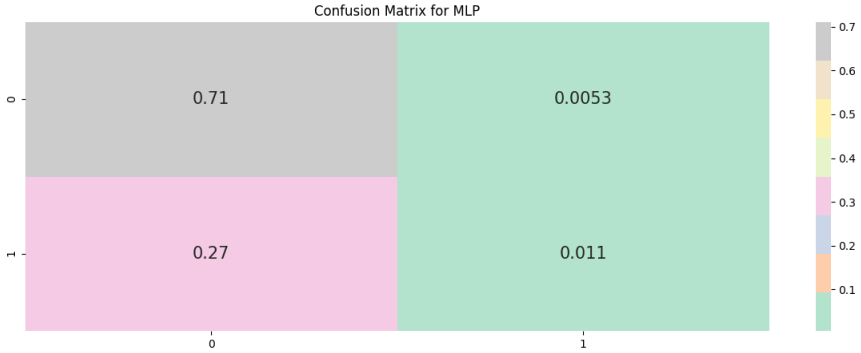
The advantages of Multi-layer Perceptron are:

- Capability to learn non-linear models.
- Capability to learn models in real-time (on-line learning).

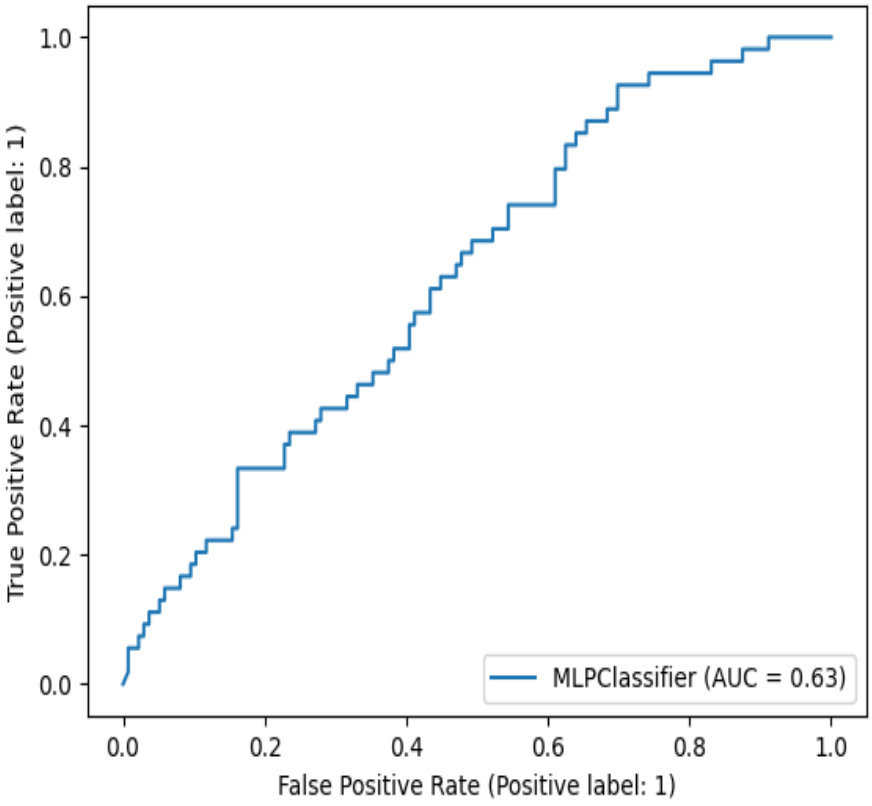
The disadvantages of Multi-layer Perceptron (MLP) include:

- MLP with hidden layers have a non-convex loss function where there exists more than one local minimum. Therefore different random weight initializations can lead to different validation accuracy.
- MLP requires tuning a number of hyperparameters such as the number of hidden neurons, layers, and iterations.
- MLP is sensitive to feature scaling.

5.8.2 Performance measures for MLP



ROC curve for Logistic Regression model with selected 5 features-
As it can be seen from plot, Area under Curve=0.63



The following is the code used in this

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.neural_network import MLPClassifier
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.model_selection import train_test_split
from sklearn import metrics
import pickle
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

# Reading dataset
df = pd.read_csv('C:/Users/nicnnnn/Documents/PCOS_Project/Datasets/PCOS_withoutinfertility.csv')

# Treating errors in dataset
del df['AMH(ng/mL)']
del df['Unnamed: 42']  # All NaN value
del df['Marraige Status (Yrs)']  # 1 NaN value
del df['Fast food (Y/N)']  # 1 NaN value

# Dropping PCOS (Y/N) column and creating target column y for predictions
X = df.drop(columns=["PCOS (Y/N)"])
y = df["PCOS (Y/N)"]

# Splitting data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.35)
```

```python
#Initializing the MLPClassifier
mlp = MLPClassifier(hidden_layer_sizes=(150,100,50), max_iter=300,activation = 'relu',solver='adam',random_state=1)

#Fitting the training data to the network
mlp.fit(X_train, y_train)

# Prediction
y_pred=mlp.predict(X_test)

# Model Evaluation
print(f"Score in Test Data : {mlp.score(X_test,y_test)}")

cm=confusion_matrix(y_test, y_pred)
p_right=cm[0][0]+cm[1][1]
p_wrong=cm[0][1]+cm[1][0]

print(f"Right classification : {p_right}")
print(f"Wrong classification : {p_wrong}")

# Finding Naive Bayes accuracy and other measures
print("MLP model accuracy(in %):", mlp.score(X_test, y_test)*100)
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))

# Generating Classification Report for NaiveBayesClassifier
classi_report = classification_report(y_test, y_pred)
print(classi_report)

# Plotting the confusion matrix for NaiveBayesClassifier
plt.subplots(figsize=(15, 5))
cf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(cf_matrix/np.sum(cf_matrix), annot=True, annot_kws={'size': 15}, cmap='Pastel2')
plt.title("Confusion Matrix for MLP")
plt.show()
```

Results and Classification Report for MLP:

```
Python - MLPclassifier.py:53 ✔

Score in Test Data : 0.7210526315789474
Right classification : 137
Wrong classification : 53
MLP model accuracy(in %): 72.10526315789474
Mean Absolute Error: 0.2789473684210526
Mean Squared Error: 0.2789473684210526
Root Mean Squared Error: 0.5281546822864043
              precision    recall  f1-score   support

           0       0.72      0.99      0.84       136
           1       0.67      0.04      0.07        54

    accuracy                           0.72       190
   macro avg       0.69      0.51      0.45       190
weighted avg       0.71      0.72      0.62       190

[Finished in 34.834s]
```

6. Clustering

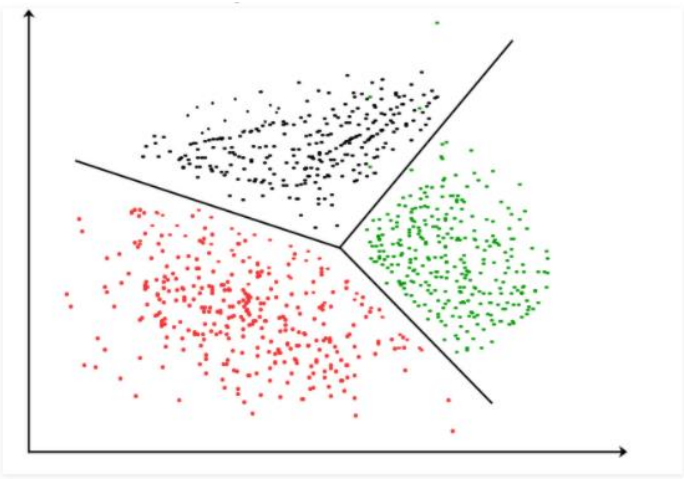
6.1 Overview of our way

Although an unsupervised machine learning technique, the clusters can be used as features in a supervised machine learning model.[12] The process of unsupervised learning has no such information yet they have some similarity with the supervised learning methodology like the requirement to work with data and the need for data exploration along with domain understanding. The difference however is that unsupervised learning does not have a target variable. This brings a whole set of problems however this is needed when one is handling data with millions or billions of instances where the output for every instance is impossible to be collected.

However, for the sake of understanding in spite of our dataset having just instances in the 100s we have attempted to use this technique to demonstrate the process of clustering. In simple language, clustering is the task of dividing data points into sections/groups in such a manner that all those data points in a group are similar. It is important to determine 'intrinsic' grouping when handling unlabeled data. There are several methods and algorithms to do this, one such technique is K-means clustering. [11]

6.2 K-means Clustering

K-means clustering is one of the simplest unsupervised learning algorithms where n observations are partitioned into k clusters. Since we can dictate the amount of clusters, it can be easily used in "classification" where we divide data into clusters which can be equal to or more than the number of classes.[12] After which the k-means algorithm can be visualized as an iterative clustering algorithm whose aim is to find the local maxima in each iteration.



Although clustering is an unsupervised machine learning approach, it can be used to improve the accuracy of supervised machine learning algorithms as well by clustering the data points into similar groups and using these cluster labels as independent variables in the supervised machine learning algorithm. [13]

6.3 Implementation

The following is the code and explanation of the implementation of KMeans clustering done by us:

- We first create a class called clust() which when initialized takes in a sklearn dataset and divides it into train and test dataset:

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.cluster import KMeans
from sklearn.metrics import accuracy_score
from sklearn import svm

class clust():
    def _load_data(self, sklearn_load_ds):
        data = sklearn_load_ds
        X = pd.DataFrame(data.data)
        self.X_train, self.X_test, self.y_train, self.y_test = train_test_split(X, data.target, test_size=0.3, random_state=42)


    def __init__(self, sklearn_load_ds):
        self._load_data(sklearn_load_ds)


    def classify(self, model=LogisticRegression(random_state=42)):
        model.fit(self.X_train, self.y_train)
        y_pred = model.predict(self.X_test)
        print('Accuracy: {}'.format(accuracy_score(self.y_test, y_pred)))


```

- Next, the function KMeans applies KMeans clustering to the train data with the number of classes as the number of clusters to be made and creates labels both for train and test data. The parameter output controls how we want to use these new labels, ‘add’ will add the labels as a feature in the dataset and ‘replace’ will use the labels instead of the train and test dataset to train our classification model.

```
    def Kmeans(self, output='add'):
        n_clusters = len(np.unique(self.y_train))
        clf = KMeans(n_clusters = n_clusters, random_state=42)
        clf.fit(self.X_train)
        y_labels_train = clf.labels_
        y_labels_test = clf.predict(self.X_test)
        if output == 'add':
            self.X_train['km_clust'] = y_labels_train
            self.X_test['km_clust'] = y_labels_test
        elif output == 'replace':
            self.X_train = y_labels_train[:, np.newaxis]
            self.X_test = y_labels_test[:, np.newaxis]
        else:
            raise ValueError('output should be either add or replace')
        return self
```

- Then we read the dataset, clean it and apply the SVM model

```
# Reading dataset
df = pd.read_csv('C:/Users/nicnnnn/Documents/PCOS_Project/Datasets/PCOS_withoutinfertility.csv')

# Treating errors in dataset
del df['AMH(ng/mL)']
del df['Unnamed: 42']  # All NaN value
del df['Marraige Status (Yrs)']  # 1 NaN value
del df['Fast food (Y/N)']  # 1 NaN value

# Dropping PCOS (Y/N) column and creating target column y for predictions
X = df.drop(columns=["PCOS (Y/N)"])
y = df["PCOS (Y/N)"]

# Splitting data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.35)

# Create SVM classifier
svc = svm.SVC(kernel='linear')
svc.fit(X_train, y_train)
```

- In the first attempt only clusters found by KMeans are used to train a classification model. These clusters alone give a decent model with an accuracy of 61.29%.

```
clust(load_digits()).Kmeans(output='replace').classify(model=svc)
```

- We compare it with the Logistic Regression model. In this case we used the features to train a Logistic Regression model. It results in a much better model with an accuracy of 96.48%.

```
clust(load_digits()).classify()
```

- Finally, we add the clusters as a feature(column) and train the same Logistic Regression model. In our final iteration we are using the clusters as features, the results show an improvement over our previous model. This gives 96.67%.

```
67   clust(load_digits()).Kmeans(output='add').classify()
```

Code output- accuracy scores for the last three instantiations:

```
Python - kMeansClustering.py:62 ✔

Accuracy: 0.6129629629629629
C:\Users\nicnnnn\AppData\Local\Prog
STOP: TOTAL NO. of ITERATIONS REACH

Increase the number of iterations (
    https://scikit-learn.org/stable
Please also refer to the documentat
    https://scikit-learn.org/stable
  n_iter_i = _check_optimize_result
Accuracy: 0.9648148148148148
C:\Users\nicnnnn\AppData\Local\Prog
STOP: TOTAL NO. of ITERATIONS REACH

Increase the number of iterations (
    https://scikit-learn.org/stable
Please also refer to the documentat
    https://scikit-learn.org/stable
  n_iter_i = _check_optimize_result
Accuracy: 0.9666666666666667
[Finished in 2.112s]
```

Hence, clustering apart from being an unsupervised machine learning can also be used to create clusters as features to improve classification models. On their own they aren't enough for classification as the results show. But when used as features they improve model accuracy.

7. Results and Conclusion

7.1 Tabulated results

Sl. no.	Model		Accuracy	Misclassification error	Precision	Recall/ tp rate	fp rate	F1 score	Area under ROC curve
1.	Gaussian Naive Bayes	All features	79.44%	20.45%	0.63	0.85	0.23	0.72	0.92
		After feature selection	88.42%	11.58%	0.78	0.92	0.1371	0.85	0.93
2.	KNN	All features	73.15%	26.85%	0.74	0.23	0.0379	0.35	0.61
		After feature selection	84.73%	15.27%	0.95	0.56	0.0164	0.71	0.91
3.	Decision Tree	All features	82.63%	17.37%	0.82	0.69	0.092	0.75	0.80
		After feature selection	87.89%	12.11%	0.84	0.77	0.0694	0.80	0.85
4.	Logistic Regression	All features	86.31%	13.69%	0.81	0.73	0.0554	0.77	0.93
		After feature selection	90.00%	10.00%	0.90	0.84	0.0619	0.87	0.94
5.	SVM	All features	89.47%	10.53%	0.86	0.80	0.0654	0.83	0.91
		After feature selection	90.52%	9.65%	0.86	0.79	0.0516	0.82	0.96
6.	Ensemble - Random Forest	All features	91.05%	8.95%	0.89	0.82	0.047	0.60	0.97
		After feature selection	92.10%	7.9%	0.87	0.88	0.0615	0.88	0.97
7.	Ensemble - AdaBoost	All features	85.78%	14.22%	0.80	0.78	0.1033	0.79	0.93
		After feature selection	88.94%	11.06%	0.75	0.91	0.1160	0.82	0.95

7.2 Inference

It can be clearly seen that the Random Forest model after feature selection has the highest accuracy of 92.10% with feature selection. We do a ranking of the classifiers on the basis of accuracy as follows:

1. Random forest (with feature selection) - 92.10%
2. Random Forest (with all features) - 91.05%
3. SVM (with feature selection) - 90.52%

4. Logistic regression (with feature selection) - 90.00%
5. SVM (with all features) - 89.47%
6. Adaboost (with feature selection) - 88.94%
7. Gaussian Naive Bayes (with feature selection) - 88.42%
8. Decision Tree (with feature selection) - 87.89%
9. Logistic regression (with all features) -86.31%
10. Adaboost (with all features) - 85.78%
11. kNN (with feature selection) - 84.73%
12. Decision Tree (with all features) - 82.63%
13. Gaussian Naive Bayes (with all features) - 79.44%
14. kNN (with all features) - 73.15%

It can be clearly seen that Random Forest has the highest accuracies for both cases of with or without feature selection. The algorithm is indeed a collection of decision trees (hence the name "forest") and it takes a "random" subset of all the existing attributes and searches for the best features among these to split a node of the tree. The algorithm performance being high and also, the highly useful "Feature Importance" visualization could be done with its use.

Next in performance, with respect to the accuracy is SVM, followed by Logistic Regression and Adaboost (ensemble classifier) as can be clearly inferred from the ranking.

There is a trade-off between Gaussian Naive Bayes and Decision Tree, so we find out the average accuracies of both (with feature selection and without feature selection and compare). The average accuracy of Gaussian Naive Bayes is 83.93% and that of Decision Tree is 85.26. So, Decision Tree is better.

Lastly we have kNN which gives a fairly poor performance as compared to all other performers, very obvious from its ROC curve.

Consequently, the Area under the ROC curve is largest for Random Forest followed by Adaboost, SVM, Logistic Regression and Gaussian Naive Bayes. Decision Tree and kNN have it on the lower side. But since all algorithms hace 0.5<AUC<1, all of them there is a high chance that all of them would distinguish the positive class values from the negative class values correctly.

The F1 scores are best for SVM, Adaboost, Logistic Regression, Decision Tree, Gaussian naive Bayes, Random Forest and kNN in that order.

So, though the Random forest model has the highest accuracy of all, it has a fairly low F1 score.

Overall, the Random Forest shows highest performance and kNN shows lowest performance.

7.3 Conclusion

As a part of the end semester project report we have firstly formulated our problem and understood the domain of this project, through extensive literature review. Next, we embarked on data exploration to understand the various features and also to analyse the statistical properties of the data. The next step was a detailed data preprocessing and data transformation using relevant techniques which included data cleaning, identification of missing values and outlier analysis. Following this we illustrated the feature selection methods and feature extraction techniques done in the lectures of this course. These form the basis of creating an efficient dataset which we can feed to the machine. Application of methods in unsupervised learning such as Clustering have been also done, catering to our binary classification problem.

The other major focus of this report has been the building and training the machine learning models using various algorithms namely Gaussian Naive Bayes, KNN, Decision Tree, Ensemble Classifier - Random Forest, Ensemble Classifier - Adaboost, Logistic Regression, SVM and Multi-layer Perceptron (Neural Network).

Of these three it is observed that the Random Forest gives best results in terms of accuracy, misclassification error and other performance measures; F1 score and area under ROC curve although cost-sensitive learning parameters like tp rate, fp rate, etc show varying performance among the other classifiers.

The final objective of this project is performing a comparative study of the various algorithms used in the training of the machine. Once trained, these machines become models and are equipped with the necessary experience which we then apply to the testing data set to gauge how well the model is trained in predicting PCOS for the various algorithms used. When a trained model encounters unseen data it should mimic the human brain in ‘generalization’ and then make a decision or prediction.

7.4 Comparison

The 2019 IEEE paper by Denny et al. had also concluded that Random Forest Classifier to give the most accurate PCOS prediction of 89.02%. [2] They had a 80-20 split of train - test dataset. They had used SPSS V22.0 DELL to identify 8 potential features based on their significance. Our project had employed similar techniques to identify top features with correlation more than 0.4 with the output. The accuracy of all models, including Random Forest saw an increase in their accuracy of predicting PCOS after feature selection. It is also to be noted that we have a 65-35 split of train - test dataset.

The 2020 IEEE Region 10 Symposium paper by Bharati et al. had concluded that a Hybrid Random Forest and Logistic Regression (RFLR) exhibited the best accuracy in predicting PCOS. They too identified 10 best features of the 43 attributed present in the given dataset resulted in better accuracy and lesser computational time. We had aptly identified this property and the benefits of 'feature selection' and have dealt with this topic in great detail throughout the report.

Hence, the observed results of our project are in line with the two IEEE reference papers which use the same data set.

The results and observations are also in line with other research papers in reputed journals using the same data set which we have referenced throughout this report.

8. Acknowledgement

We would like to thank Dr. Madan Gopal for introducing us to the field of Machine Learning through his EED363 lectures and for his constant encouragement and guidance for us to bravely explore and understand the various concepts that form the bedrock of this field. We are also incredibly grateful for his book 'Applied Machine Learning' which has proved to be an invaluable resource for us in the process of completing this report. We would also like to thank the entire team of Mr. Venugopala Kotha and student teaching assistants for their constant guidance during the course and the preparation of the midterm and final reports. Lastly, we would like to express our gratitude to our seniors and alumni members for their guidance and expertise towards the completion of this project.

9. References:

[1] McCartney, C.R. and Marshall, J.C., 2016. Polycystic ovary syndrome. New England Journal of Medicine, 375(1), pp.54-64.

[2] A. Denny, A. Raj, A. Ashok, C. M. Ram and R. George, "i-HOPE: Detection And Prediction System For Polycystic Ovary Syndrome (PCOS) Using Machine Learning Techniques," TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON), Kochi, India, 2019, pp. 673-678.

[3] S. Bharati, P. Podder, M. R. H. Mondal, " Diagnosis of Polycystic Ovary Syndrome Using Machine Learning Algorithms", TENSYMP 2020 - 2020 IEEE Region 10 Symposium (TENSYMP), 2020, pp. 1486 - 1489.

[4] A.Munjal, R. Khandia, B. Gautam, "A Machine Learning Approach for Selection of Polycystic Ovarian Syndrome (PCOS) Attributes and comparing different classifier performance with the help of weka and pycaret" Volume - 9, Issue - 12, December 2020.

[5] M. H. Hassan, T. Mirza, " Comparative Analysis of Machine Learning Algorithms in Diagnosis of Polycystic Ovarian Syndrome", International Journal of Computer Applications, Volume 175 - No. 17, September 2020.

[6] N. Tanwani, "Detecting PCOS using Machine Learning", International Journal of Modern Trends in Engineering and Science", Volume: 07 Issue 01 2020.

[7] Gopal, Madan. "Applied Machine Learning", McGraw Hill Education (india), 2018

[8] Koehrsen, Will. "An Implementation and Explanation of the Random Forest in Python" 2018. Last accessed athttps://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76 on 02.02.2021

[9] Brownlee, Jason. "Logistic Regression for Machine Learning" 2020. Last accessed at https://machinelearningmastery.com/logistic-regression-for-machine-learning/ on 08.04.2021.

[10] Shaikh, Raheel. "Feature Selection Techniques in Machine Learning with Python" 2018 Last accessed at https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e on 12/05/2021.

[11] GeeksforGeeks. "Clustering in Machine Learning" 2020. Last accessed at https://www.geeksforgeeks.org/clustering-in-machine-learning/ on 01/05/2021.

[12] Khan, Mudassir. “KMeans Clustering for Classification” 2017. Last accessed at https://towardsdatascience.com/kmeans-clustering-for-classification-74b992405d0a on 01/05/2021.

[13] Kaushik, Saurav. “An Introduction to Clustering and different methods of clustering” 2016. Last accessed at https://www.analyticsvidhya.com/blog/2016/11/an-introduction-to
-clustering-and-different-methods-of-clustering/ on 30/04/2021.

[14] Ippolito, Pier. “Feature Extraction Techniques” 2019. Last accessed at https://towardsdatascience.com/feature-extraction-techniques-d619b56e31be on 21/04/2021.

[15] Zhou, Victor. “Machine Learning for Beginners: An Introduction to neural networks” 2019. Last accessed at https://towardsdatascience.com/machine-learning-For-beginners-an-introduction-to-neural-networks-d49f22d238f9 on 14/04/2021.

[16] Algorithmia. “Feature Selection in machine learning” Last accessed at https://algorithmia.com/blog/feature-selection-in-machine-learning on 29/03/2021.