Data Pre-processing

In [415…

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

In [416…

```
from sklearn import preprocessing as ps
from sklearn import metrics
from sklearn.model_selection import RepeatedKFold , StratifiedKFold
from sklearn.preprocessing import StandardScaler as sc
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, ConfusionMatrixDisplay
from sklearn.model_selection import KFold
```

In [417…

```
from sklearn import tree
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from xgboost import XGBClassifier
```

In [418…

```
pcos_dt = pd.read_csv('PCOS_data_without_infertility.csv')
pcos_dt.head()
```

Out[418]:

	Sl. No	Patient File No.	PCOS (Y/N)	Age (yrs)	Weight (Kg)	Height(Cm)	BMI	Blood Group	Pulse rate(bpm)	RR (breaths/min)	...	Fast food (Y/N)	Reg.
0	1	1	0	28	44.6	152.0	19.3	15	78	22	...	1.0	
1	2	2	0	36	65.0	161.5	#NAME?	15	74	20	...	0.0	
2	3	3	1	33	68.8	165.0	#NAME?	11	72	18	...	1.0	
3	4	4	0	37	65.0	148.0	#NAME?	13	72	20	...	0.0	
4	5	5	0	25	52.0	161.0	#NAME?	11	72	18	...	0.0	

5 rows × 44 columns

In [419…

```
pcos_dt.shape
```

Out[419]:

```
(541, 44)
```

In [420…

```
pcos_dt.columns
```

Out[420]:

```
Index(['Sl. No', 'Patient File No.', 'PCOS (Y/N)', ' Age (yrs)', 'Weight (Kg)',
       'Height(Cm) ', 'BMI', 'Blood Group', 'Pulse rate(bpm) ',
       'RR (breaths/min)', 'Hb(g/dl)', 'Cycle(R/I)', 'Cycle length(days)',
       'Pregnant(Y/N)', 'No. of aborptions', '  I   beta-HCG(mIU/mL)',
       'II    beta-HCG(mIU/mL)', 'FSH(mIU/mL)', 'LH(mIU/mL)', 'FSH/LH',
       'Hip(inch)', 'Waist(inch)', 'Waist:Hip Ratio', 'TSH (mIU/L)',
       'AMH(ng/mL)', 'PRL(ng/mL)', 'Vit D3 (ng/mL)', 'PRG(ng/mL)',
       'RBS(mg/dl)', 'Weight gain(Y/N)', 'hair growth(Y/N)',
       'Skin darkening (Y/N)', 'Hair loss(Y/N)', 'Pimples(Y/N)',
       'Fast food (Y/N)', 'Reg.Exercise(Y/N)', 'BP _Systolic (mmHg)',
       'BP _Diastolic (mmHg)', 'Follicle No. (L)', 'Follicle No. (R)',
       'Avg. F size (L) (mm)', 'Avg. F size (R) (mm)', 'Endometrium (mm)',
       'Unnamed: 43'],
      dtype='object')
```

In [421…

```
#To remove whitespaces at both ends from a column name

pcos_dt.columns = pcos_dt.columns.str.strip()
```

In [422…

```
pcos_dt.columns
```

Out[422]:

```
Index(['Sl. No', 'Patient File No.', 'PCOS (Y/N)', 'Age (yrs)', 'Weight (Kg)',
       'Height(Cm)', 'BMI', 'Blood Group', 'Pulse rate(bpm)',
       'RR (breaths/min)', 'Hb(g/dl)', 'Cycle(R/I)', 'Cycle length(days)',
       'Pregnant(Y/N)', 'No. of aborptions', 'I   beta-HCG(mIU/mL)',
       'II    beta-HCG(mIU/mL)', 'FSH(mIU/mL)', 'LH(mIU/mL)', 'FSH/LH',
       'Hip(inch)', 'Waist(inch)', 'Waist:Hip Ratio', 'TSH (mIU/L)',
       'AMH(ng/mL)', 'PRL(ng/mL)', 'Vit D3 (ng/mL)', 'PRG(ng/mL)',
       'RBS(mg/dl)', 'Weight gain(Y/N)', 'hair growth(Y/N)',
       'Skin darkening (Y/N)', 'Hair loss(Y/N)', 'Pimples(Y/N)',
       'Fast food (Y/N)', 'Reg.Exercise(Y/N)', 'BP _Systolic (mmHg)',
       'BP _Diastolic (mmHg)', 'Follicle No. (L)', 'Follicle No. (R)',
       'Avg. F size (L) (mm)', 'Avg. F size (R) (mm)', 'Endometrium (mm)',
       'Unnamed: 43'],
      dtype='object')
```

Removing unwanted columns

In [423…

```
pcos_dt.drop(['Sl. No', 'Patient File No.','Unnamed: 43'],axis='columns',inplace=True)
```

Imputing Missing values

Rows with unmatching values are removed and some values are replaced

In [424…

```
pcos_dt = pcos_dt.replace(pcos_dt['II    beta-HCG(mIU/mL)'][123],'1.99')
pcos_dt.drop(labels=305,axis=0,inplace=True)
```

Searching for columns with missing values

In [425…

```
pcos_dt.isnull().sum()
```

Out[425]:

```
PCOS (Y/N)                0
Age (yrs)                 0
Weight (Kg)               0
Height(Cm)                0
BMI                       0
Blood Group               0
Pulse rate(bpm)           0
RR (breaths/min)          0
Hb(g/dl)                  0
Cycle(R/I)                0
Cycle length(days)        0
Pregnant(Y/N)             0
No. of aborptions         0
I   beta-HCG(mIU/mL)      0
II    beta-HCG(mIU/mL)    0
FSH(mIU/mL)               0
LH(mIU/mL)                0
FSH/LH                    0
Hip(inch)                 0
Waist(inch)               0
Waist:Hip Ratio           0
TSH (mIU/L)               0
AMH(ng/mL)                0
PRL(ng/mL)                0
Vit D3 (ng/mL)            0
PRG(ng/mL)                0
RBS(mg/dl)                0
Weight gain(Y/N)          0
hair growth(Y/N)          0
Skin darkening (Y/N)      0
Hair loss(Y/N)            0
Pimples(Y/N)              0
Fast food (Y/N)           1
Reg.Exercise(Y/N)         0
BP _Systolic (mmHg)       0
BP _Diastolic (mmHg)      0
Follicle No. (L)          0
Follicle No. (R)          0
Avg. F size (L) (mm)      0
Avg. F size (R) (mm)      0
Endometrium (mm)          0
dtype: int64
```

In [426…

```
# Replacing the missing values in a feature column with the median of the feature
pcos_dt['Fast food (Y/N)'].fillna(pcos_dt['Fast food (Y/N)'].median(), inplace = True)
```

In [427…

```
pcos_dt.isnull().sum()
```

Out[427]:

```
PCOS (Y/N)                0
Age (yrs)                 0
Weight (Kg)               0
Height(Cm)                0
BMI                       0
Blood Group               0
Pulse rate(bpm)           0
RR (breaths/min)          0
Hb(g/dl)                  0
Cycle(R/I)                0
Cycle length(days)        0
Pregnant(Y/N)             0
No. of aborptions         0
I   beta-HCG(mIU/mL)      0
II    beta-HCG(mIU/mL)    0
FSH(mIU/mL)               0
LH(mIU/mL)                0
FSH/LH                    0
Hip(inch)                 0
Waist(inch)               0
Waist:Hip Ratio           0
TSH (mIU/L)               0
AMH(ng/mL)                0
PRL(ng/mL)                0
Vit D3 (ng/mL)            0
PRG(ng/mL)                0
RBS(mg/dl)                0
Weight gain(Y/N)          0
hair growth(Y/N)          0
Skin darkening (Y/N)      0
Hair loss(Y/N)            0
Pimples(Y/N)              0
Fast food (Y/N)           0
Reg.Exercise(Y/N)         0
BP _Systolic (mmHg)       0
BP _Diastolic (mmHg)      0
Follicle No. (L)          0
Follicle No. (R)          0
Avg. F size (L) (mm)      0
Avg. F size (R) (mm)      0
Endometrium (mm)          0
dtype: int64
```

Standardization

In [428…

```
pcos_dt = pcos_dt.replace('#NAME?', np.nan)
pcos_dt = pcos_dt.apply(pd.to_numeric, errors='coerce')
pcos_dt = pcos_dt.fillna(pcos_dt.mean())  # You can also use median() or other imputation met
```

In [429…

```
pcos_dt.head()
```

Out[429]:

	PCOS (Y/N)	Age (yrs)	Weight (Kg)	Height(Cm)	BMI	Blood Group	Pulse rate(bpm)	RR (breaths/min)	Hb(g/dl)	Cycle(R/I)	...	Pi
0	0	28	44.6	152.0	19.300000	15	78	22	10.48	2	...	
1	0	36	65.0	161.5	23.929752	15	74	20	11.70	2	...	
2	1	33	68.8	165.0	23.929752	11	72	18	11.80	2	...	
3	0	37	65.0	148.0	23.929752	13	72	20	12.00	2	...	
4	0	25	52.0	161.0	23.929752	11	72	18	10.00	2	...	

5 rows × 41 columns

Normalization

In [430…

```
scaler = ps.MinMaxScaler()
pcos_sc = scaler.fit_transform(pcos_dt)
pcos_normalized_dt = pd.DataFrame(pcos_sc,columns = pcos_dt.columns)
```

In [431…

```
pcos_normalized_dt.head()
```

Out[431]:

	PCOS (Y/N)	Age (yrs)	Weight (Kg)	Height(Cm)	BMI	Blood Group	Pulse rate(bpm)	RR (breaths/min)	Hb(g/dl)	Cycle(R/I)
0	0.0	0.285714	0.176623	0.348837	0.176471	0.571429	0.942029	0.500000	0.314286	0.0
1	0.0	0.571429	0.441558	0.569767	0.370998	0.571429	0.884058	0.333333	0.507937	0.0
2	1.0	0.464286	0.490909	0.651163	0.370998	0.000000	0.855072	0.166667	0.523810	0.0
3	0.0	0.607143	0.441558	0.255814	0.370998	0.285714	0.855072	0.333333	0.555556	0.0
4	0.0	0.178571	0.272727	0.558140	0.370998	0.000000	0.855072	0.166667	0.238095	0.0

5 rows × 41 columns

Data Summarization

In [432…

```
pcos_dt.shape
```

Out[432]:

```
(540, 41)
```

In [433…

```
pcos_dt.dtypes
```

Out[433]:

```
PCOS (Y/N)                  int64
Age (yrs)                   int64
Weight (Kg)               float64
Height(Cm)                float64
BMI                       float64
Blood Group                 int64
Pulse rate(bpm)             int64
RR (breaths/min)            int64
Hb(g/dl)                  float64
Cycle(R/I)                  int64
Cycle length(days)          int64
Pregnant(Y/N)               int64
No. of aborptions           int64
I   beta-HCG(mIU/mL)      float64
II    beta-HCG(mIU/mL)    float64
FSH(mIU/mL)               float64
LH(mIU/mL)                float64
FSH/LH                    float64
Hip(inch)                   int64
Waist(inch)                 int64
Waist:Hip Ratio           float64
TSH (mIU/L)               float64
AMH(ng/mL)                float64
PRL(ng/mL)                float64
Vit D3 (ng/mL)            float64
PRG(ng/mL)                float64
RBS(mg/dl)                float64
Weight gain(Y/N)            int64
hair growth(Y/N)            int64
Skin darkening (Y/N)        int64
Hair loss(Y/N)              int64
Pimples(Y/N)                int64
Fast food (Y/N)           float64
Reg.Exercise(Y/N)           int64
BP _Systolic (mmHg)         int64
BP _Diastolic (mmHg)        int64
Follicle No. (L)            int64
Follicle No. (R)            int64
Avg. F size (L) (mm)      float64
Avg. F size (R) (mm)      float64
Endometrium (mm)          float64
dtype: object
```

In [434…

```
pcos_dt.describe()
```

Out[434]:

	PCOS (Y/N)	Age (yrs)	Weight (Kg)	Height(Cm)	BMI	Blood Group	Pulse rate(bpm)	RR (breaths/min)
count	540.000000	540.000000	540.000000	540.000000	540.000000	540.000000	540.000000	540.000000
mean	0.327778	31.420370	59.643889	156.493141	23.929752	13.803704	73.246296	19.242593
std	0.469839	5.410698	11.037399	6.036043	2.449469	1.842194	4.434274	1.689881
min	0.000000	20.000000	31.000000	137.000000	15.100000	11.000000	13.000000	16.000000
25%	0.000000	27.750000	52.000000	152.000000	23.929752	13.000000	72.000000	18.000000
50%	0.000000	31.000000	59.300000	156.000000	23.929752	14.000000	72.000000	18.000000
75%	1.000000	35.000000	65.000000	160.000000	23.929752	15.000000	74.000000	20.000000
max	1.000000	48.000000	108.000000	180.000000	38.900000	18.000000	82.000000	28.000000

8 rows × 41 columns

In [435…

```
pcos_dt.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 540 entries, 0 to 540
Data columns (total 41 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   PCOS (Y/N)              540 non-null    int64  
 1    Age (yrs)              540 non-null    int64  
 2   Weight (Kg)             540 non-null    float64
 3   Height(Cm)              540 non-null    float64
 4   BMI                     540 non-null    float64
 5   Blood Group             540 non-null    int64  
 6   Pulse rate(bpm)         540 non-null    int64  
 7   RR (breaths/min)        540 non-null    int64  
 8   Hb(g/dl)                540 non-null    float64
 9   Cycle(R/I)              540 non-null    int64  
 10  Cycle length(days)      540 non-null    int64  
 11  Pregnant(Y/N)           540 non-null    int64  
 12  No. of aborptions       540 non-null    int64  
 13    I   beta-HCG(mIU/mL)  540 non-null    float64
 14  II    beta-HCG(mIU/mL)  540 non-null    float64
 15  FSH(mIU/mL)             540 non-null    float64
 16  LH(mIU/mL)              540 non-null    float64
 17  FSH/LH                  540 non-null    float64
 18  Hip(inch)               540 non-null    int64  
 19  Waist(inch)             540 non-null    int64  
 20  Waist:Hip Ratio         540 non-null    float64
 21  TSH (mIU/L)             540 non-null    float64
 22  AMH(ng/mL)              540 non-null    float64
 23  PRL(ng/mL)              540 non-null    float64
 24  Vit D3 (ng/mL)          540 non-null    float64
 25  PRG(ng/mL)              540 non-null    float64
 26  RBS(mg/dl)              540 non-null    float64
 27  Weight gain(Y/N)        540 non-null    int64  
 28  hair growth(Y/N)        540 non-null    int64  
 29  Skin darkening (Y/N)    540 non-null    int64  
 30  Hair loss(Y/N)          540 non-null    int64  
 31  Pimples(Y/N)            540 non-null    int64  
 32  Fast food (Y/N)         540 non-null    float64
 33  Reg.Exercise(Y/N)       540 non-null    int64  
 34  BP _Systolic (mmHg)     540 non-null    int64  
 35  BP _Diastolic (mmHg)    540 non-null    int64  
 36  Follicle No. (L)        540 non-null    int64  
 37  Follicle No. (R)        540 non-null    int64  
 38  Avg. F size (L) (mm)    540 non-null    float64
 39  Avg. F size (R) (mm)    540 non-null    float64
 40  Endometrium (mm)        540 non-null    float64
dtypes: float64(20), int64(21)
memory usage: 177.2 KB
```

In [436…

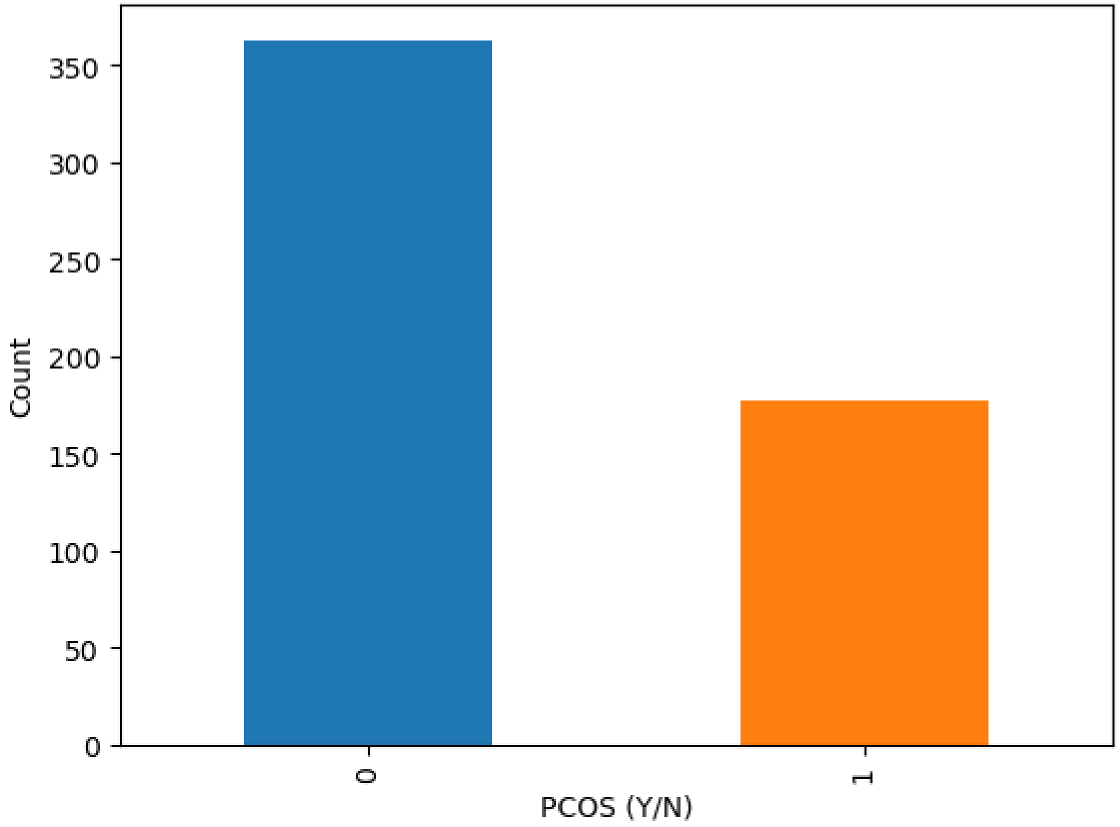
```
pcos_dt.to_csv("pcos_datatset_cleaned1.csv")
```

Data Visualization

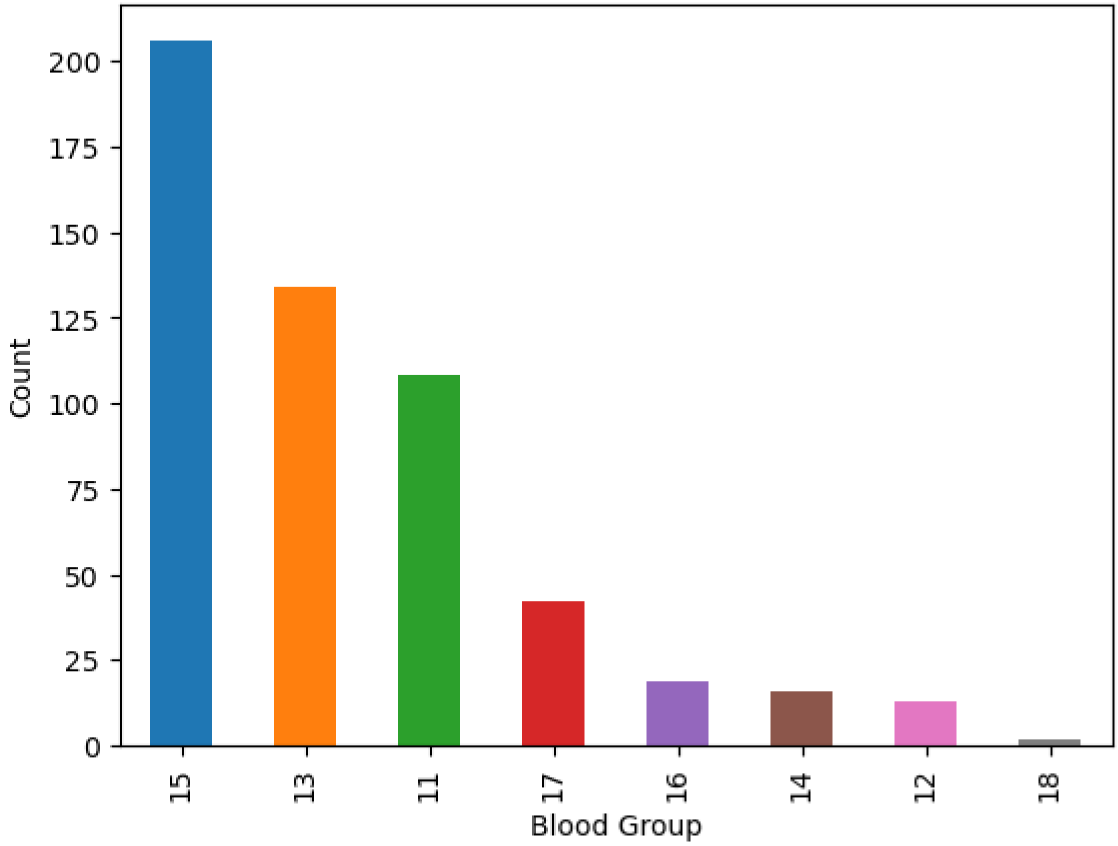
In [437…

```
# Plotting all the categorical variables using bar plot
cv = ['PCOS (Y/N)','Blood Group','Pregnant(Y/N)','Weight gain(Y/N)','hair growth(Y/N)','Skin
      'Fast food (Y/N)', 'Reg.Exercise(Y/N)']

for i in cv:
  plt.xlabel(i)
  plt.ylabel('Count')
  c=pcos_dt[i].value_counts()
  print(i)
  print(c)
  colors = [('C'+str(j)) for j in range(len(c))]
  pcos_dt[i].value_counts().plot(kind='bar',color=colors)
  plt.show()
```

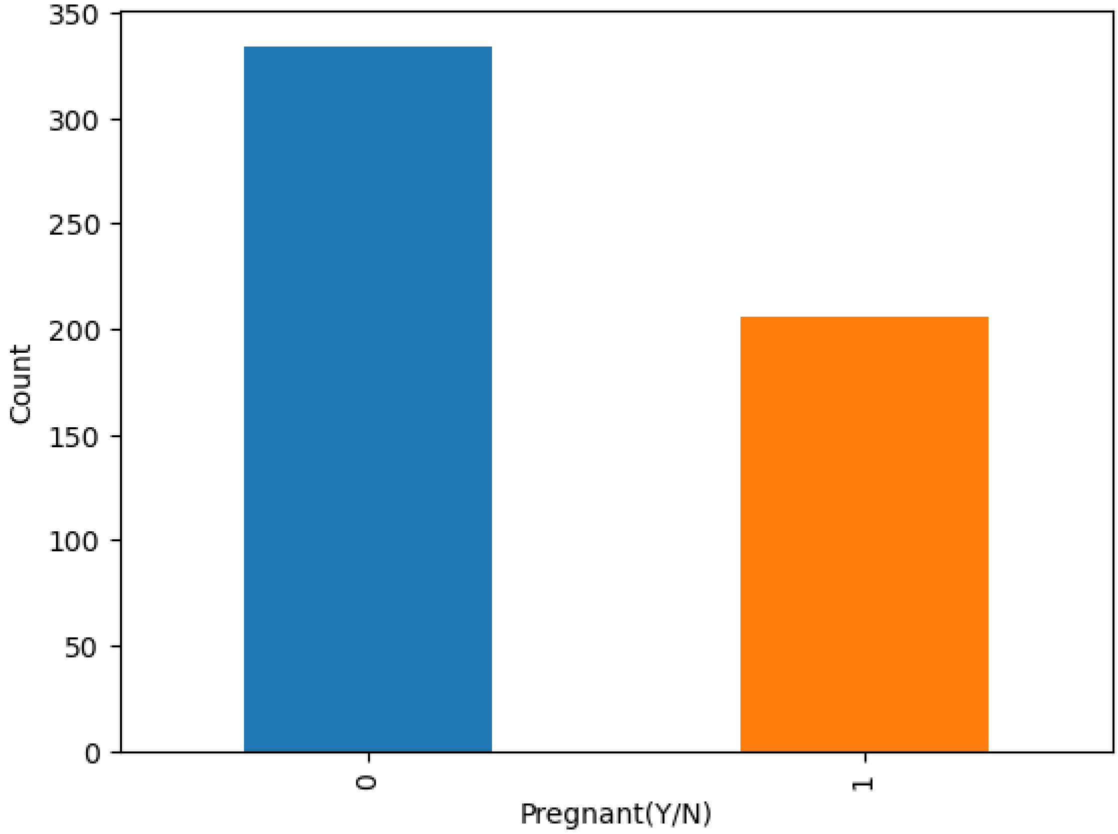
```
PCOS (Y/N)
PCOS (Y/N)
0    363
1    177
Name: count, dtype: int64
```



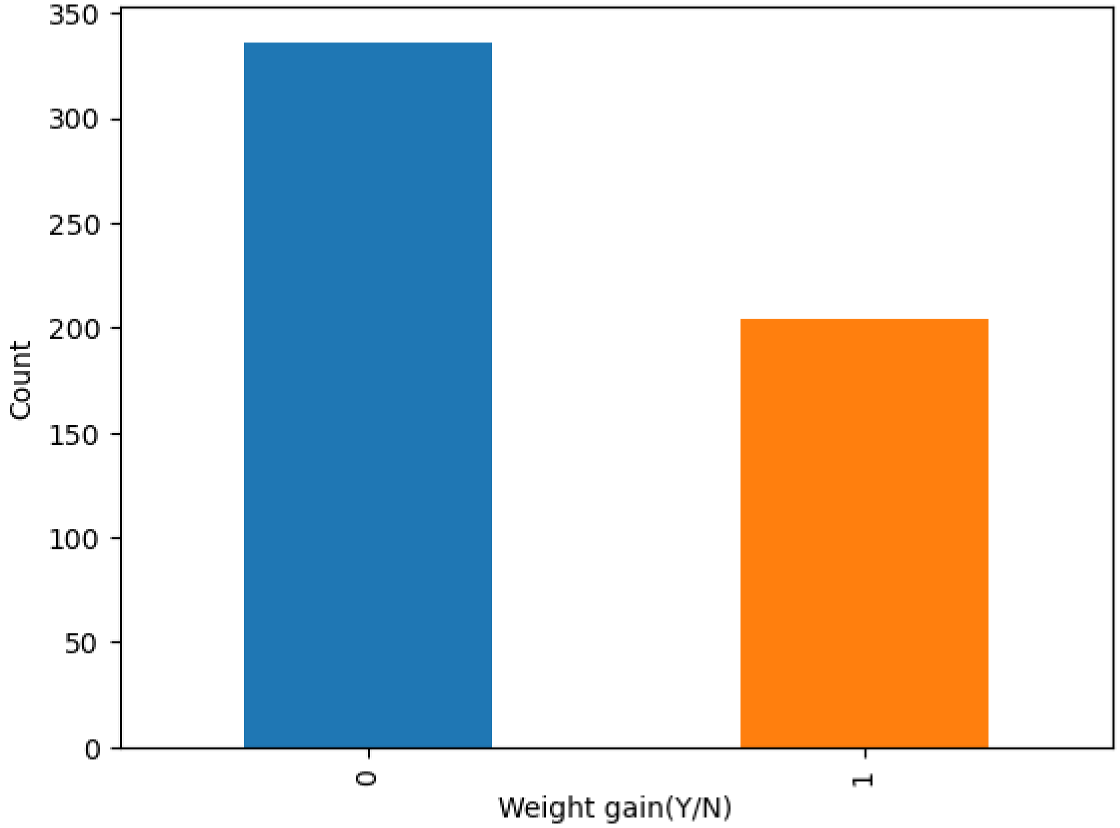
```
Blood Group
Blood Group
15    206
13    134
11    108
17     42
16     19
14     16
12     13
18      2
Name: count, dtype: int64
```



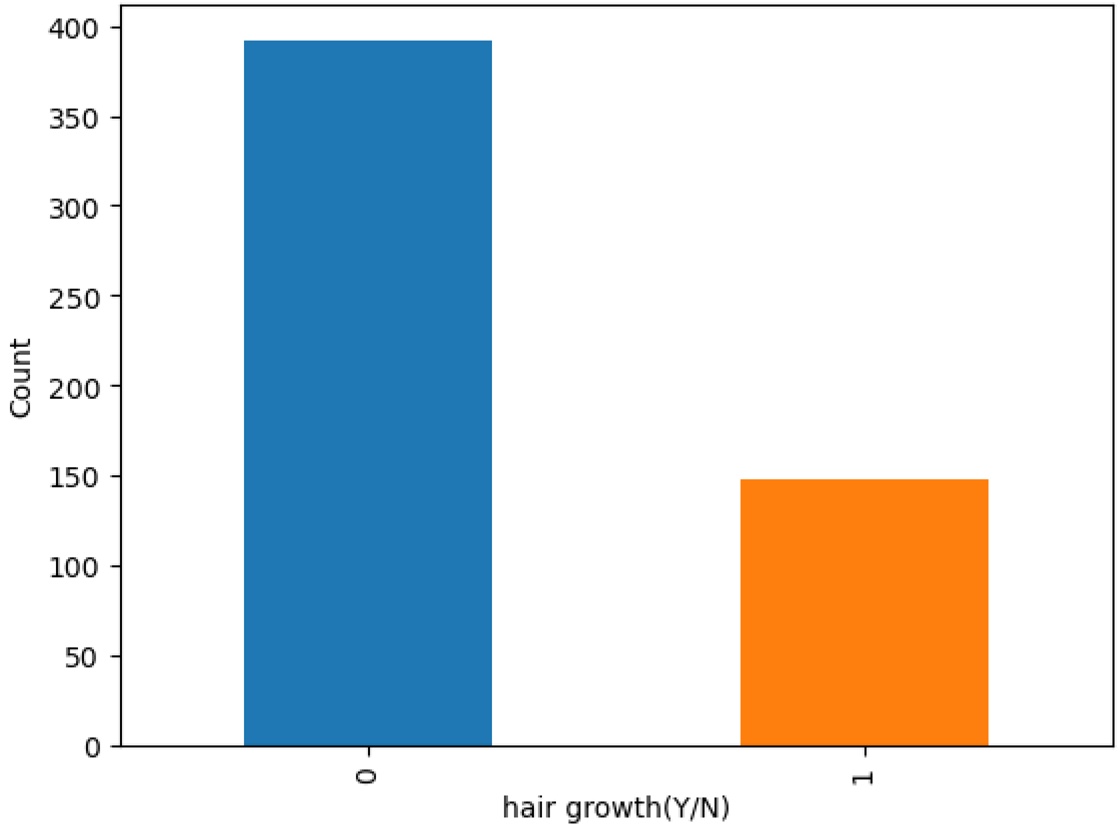
```
Pregnant(Y/N)
Pregnant(Y/N)
0    334
1    206
Name: count, dtype: int64
```



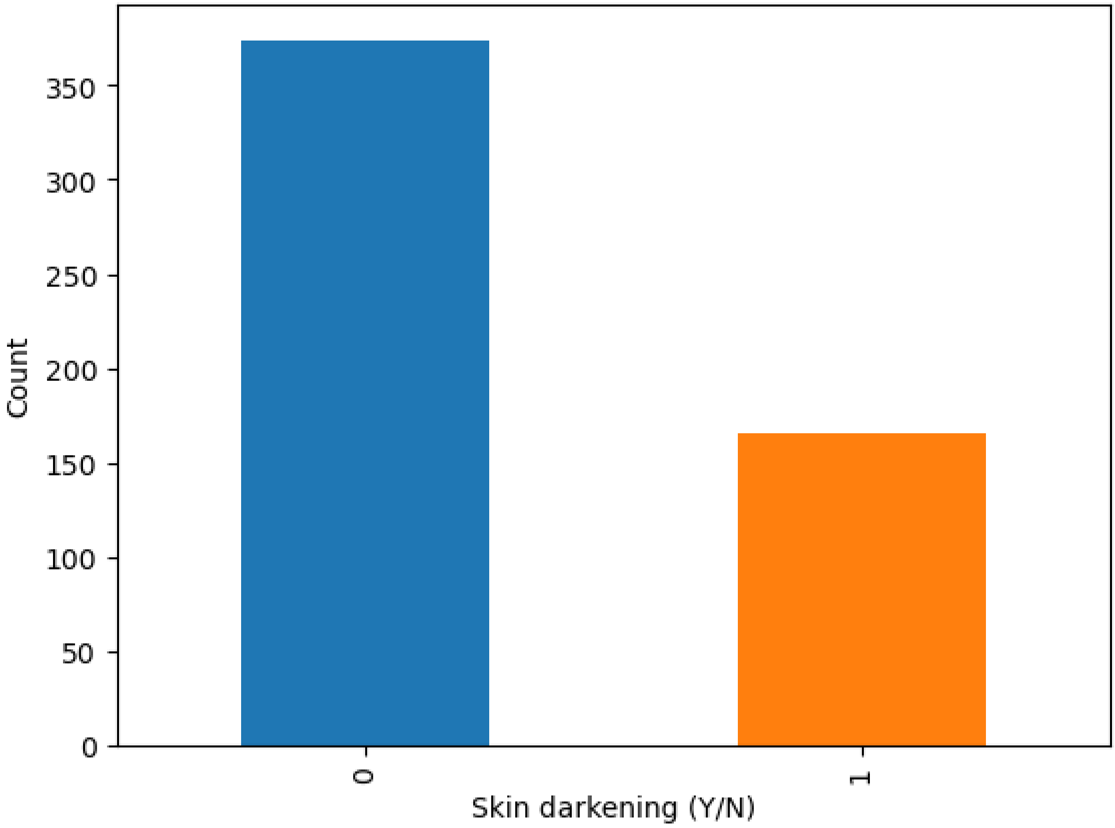
```
Weight gain(Y/N)
Weight gain(Y/N)
0    336
1    204
Name: count, dtype: int64
```



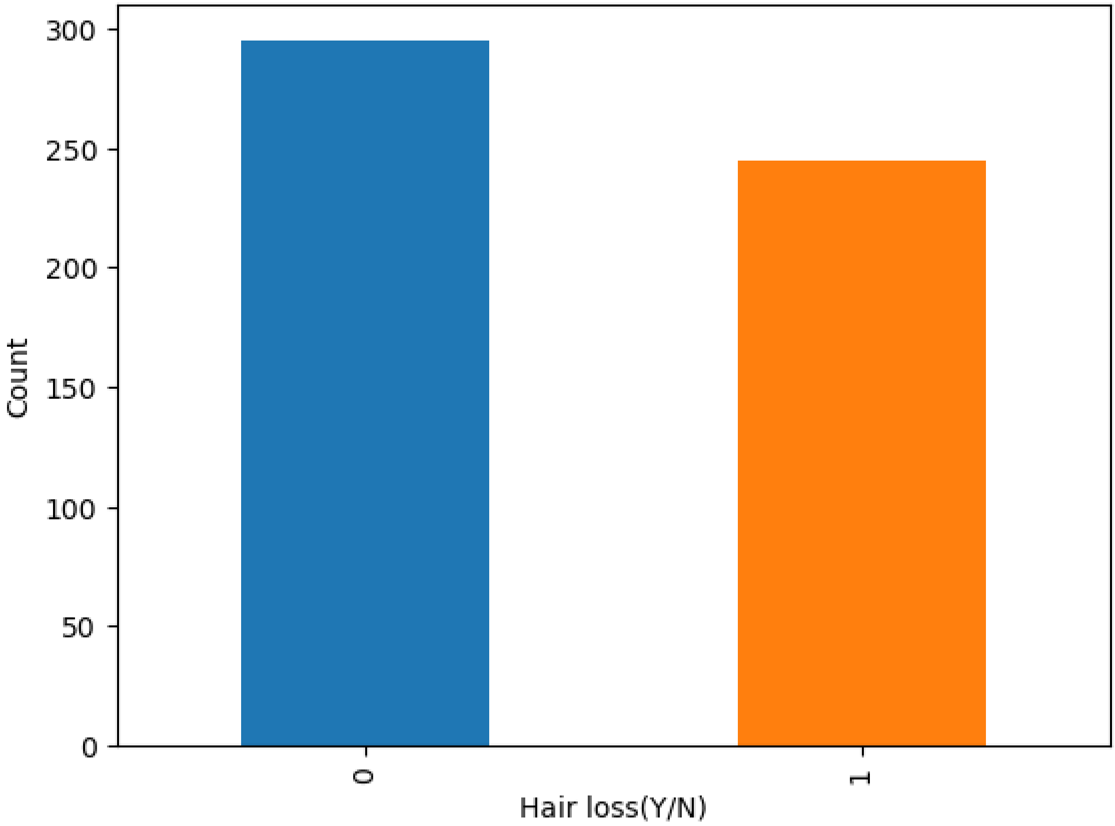
```
hair growth(Y/N)
hair growth(Y/N)
0    392
1    148
Name: count, dtype: int64
```



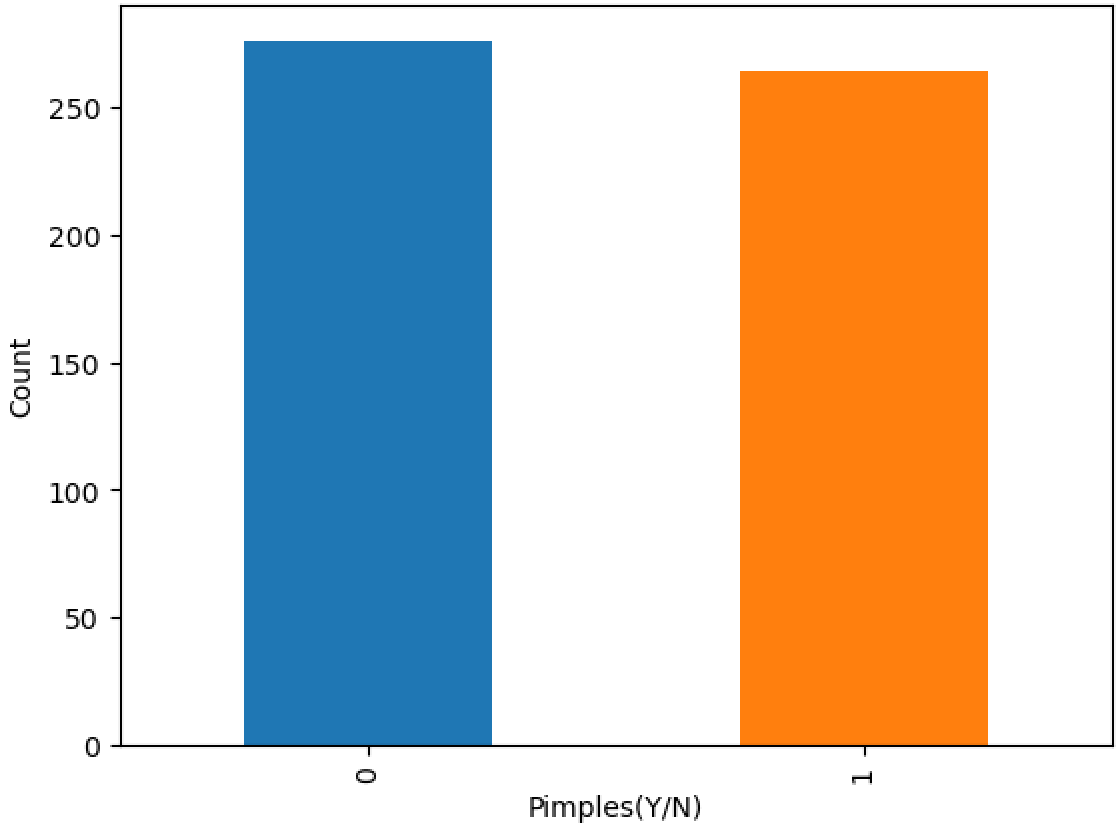
```
Skin darkening (Y/N)
Skin darkening (Y/N)
0    374
1    166
Name: count, dtype: int64
```



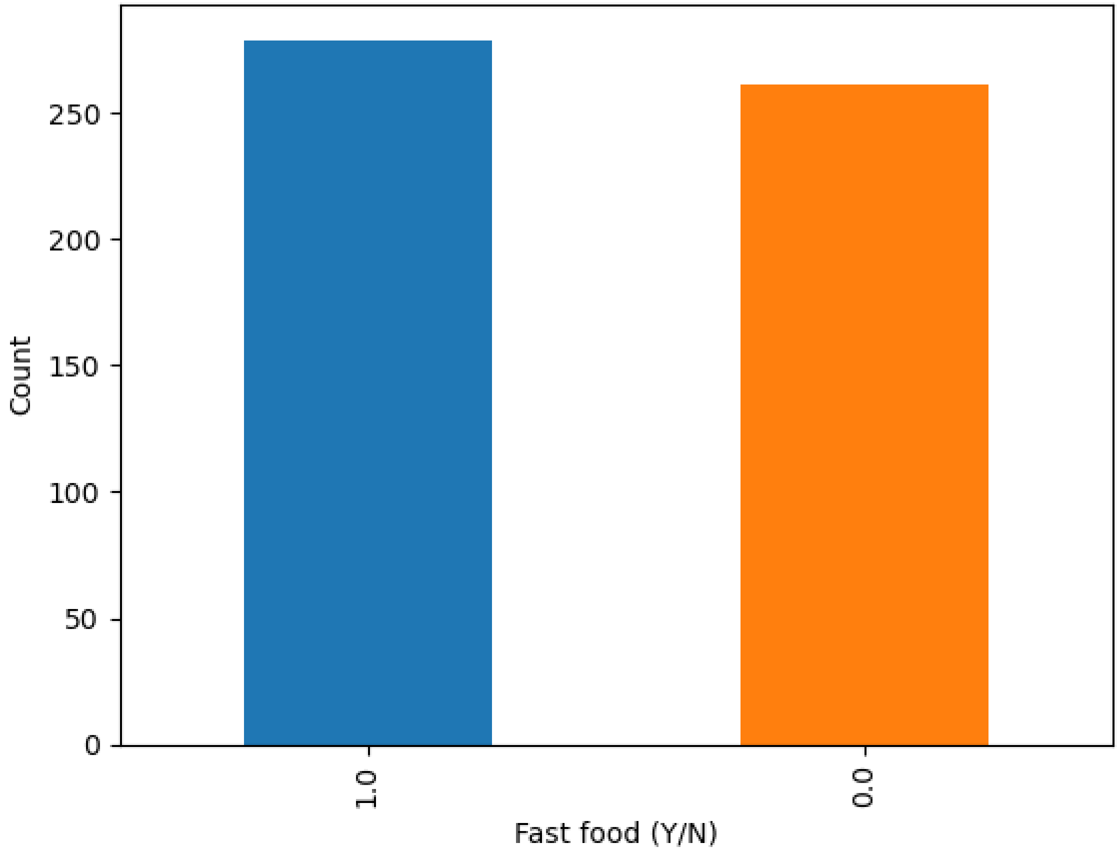
```
Hair loss(Y/N)
Hair loss(Y/N)
0    295
1    245
Name: count, dtype: int64
```



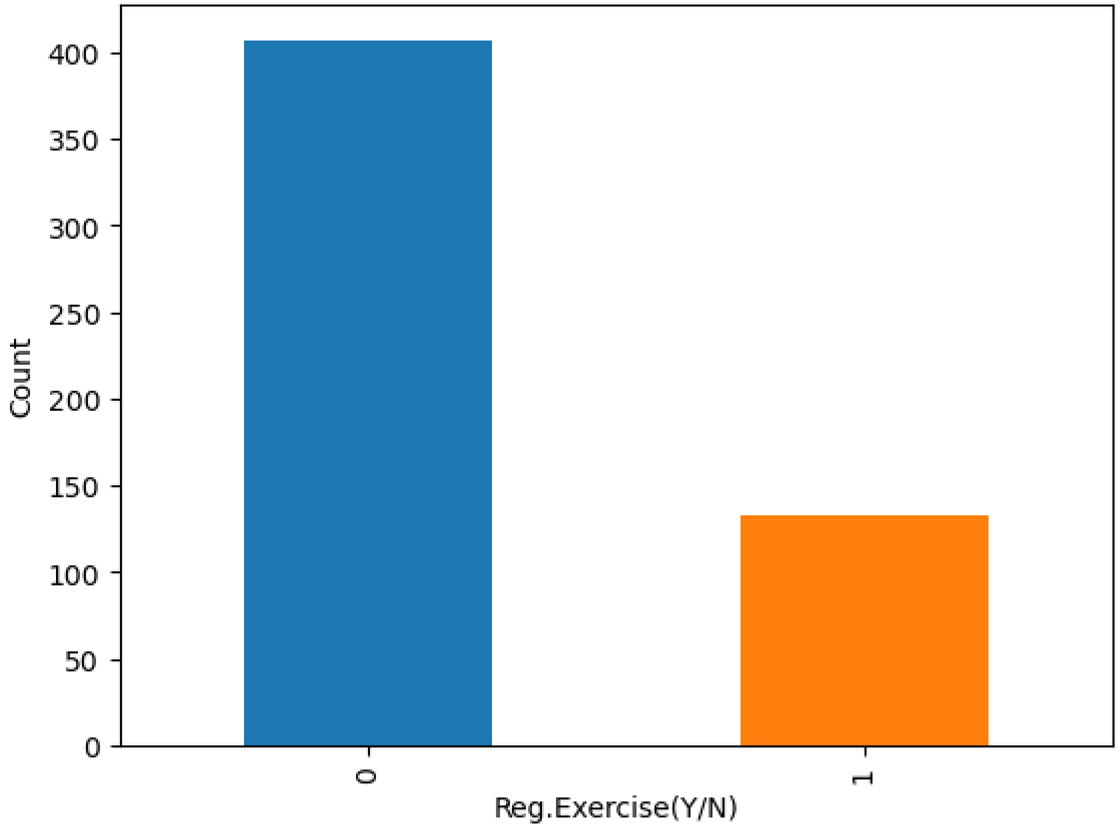
```
Pimples(Y/N)
Pimples(Y/N)
0    276
1    264
Name: count, dtype: int64
```



```
Fast food (Y/N)
Fast food (Y/N)
1.0    279
0.0    261
Name: count, dtype: int64
```

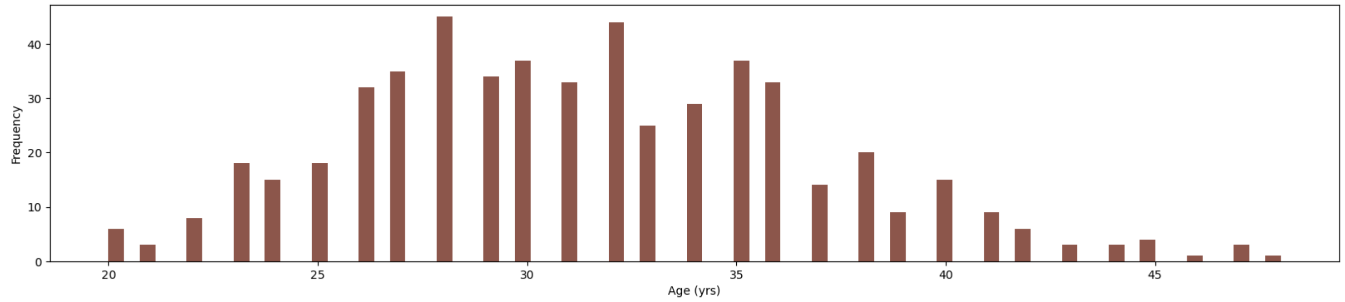


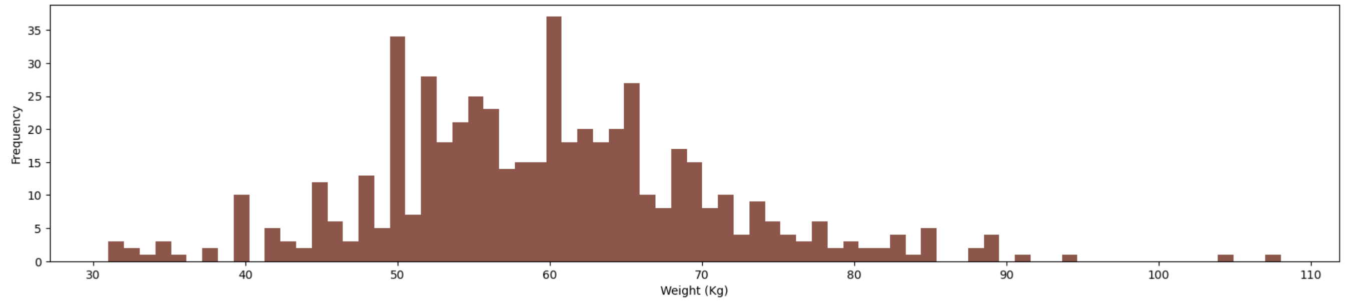
```
Reg.Exercise(Y/N)
Reg.Exercise(Y/N)
0    407
1    133
Name: count, dtype: int64
```

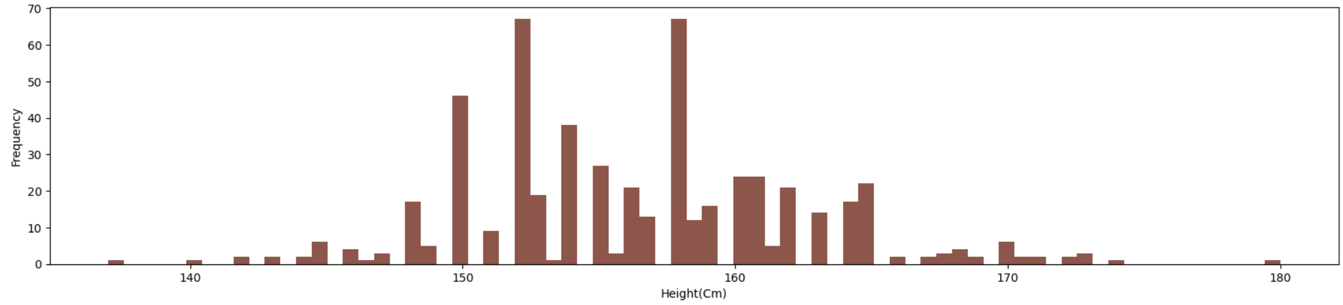


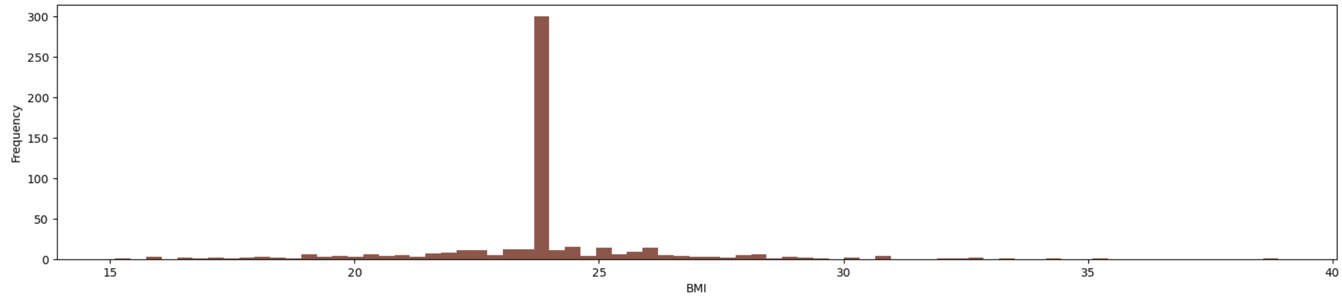
In [438…

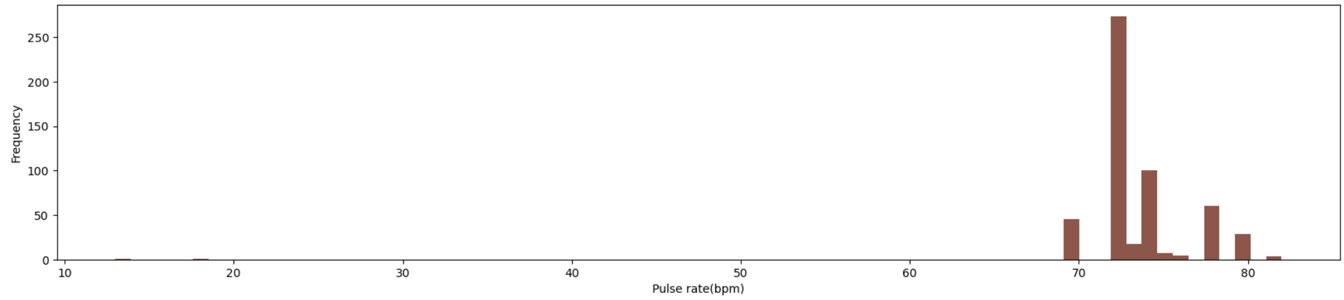
```
# Plotting all the numerical variables using bar plot
nv = ['Age (yrs)', 'Weight (Kg)','Height(Cm)', 'BMI','Pulse rate(bpm)',
       'RR (breaths/min)', 'Hb(g/dl)', 'Cycle(R/I)', 'Cycle length(days)',
       'I   beta-HCG(mIU/mL)', 'FSH(mIU/mL)',
       'LH(mIU/mL)', 'FSH/LH', 'Hip(inch)', 'Waist(inch)', 'Waist:Hip Ratio',
       'TSH (mIU/L)', 'PRL(ng/mL)', 'Vit D3 (ng/mL)',
       'PRG(ng/mL)', 'RBS(mg/dl)','BP _Systolic (mmHg)',
       'BP _Diastolic (mmHg)', 'Follicle No. (L)', 'Follicle No. (R)',
       'Avg. F size (L) (mm)', 'Avg. F size (R) (mm)', 'Endometrium (mm)']
for i in nv:
  plt.figure(figsize=(20,4))
  plt.hist(pcos_dt[i], bins = 75,color='C5')
  plt.xlabel(i)
  plt.ylabel('Frequency')
  plt.show()
```

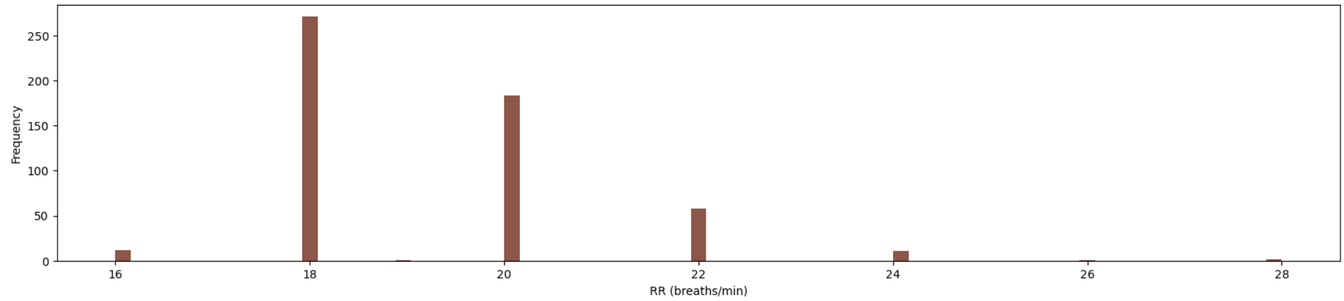


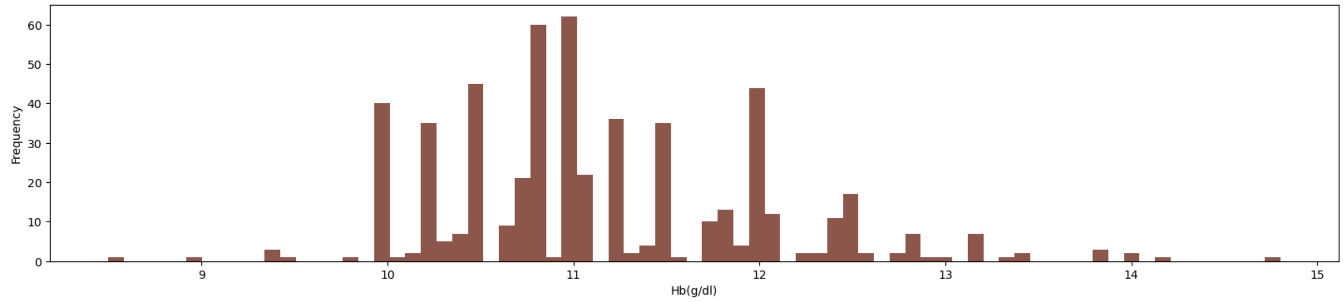


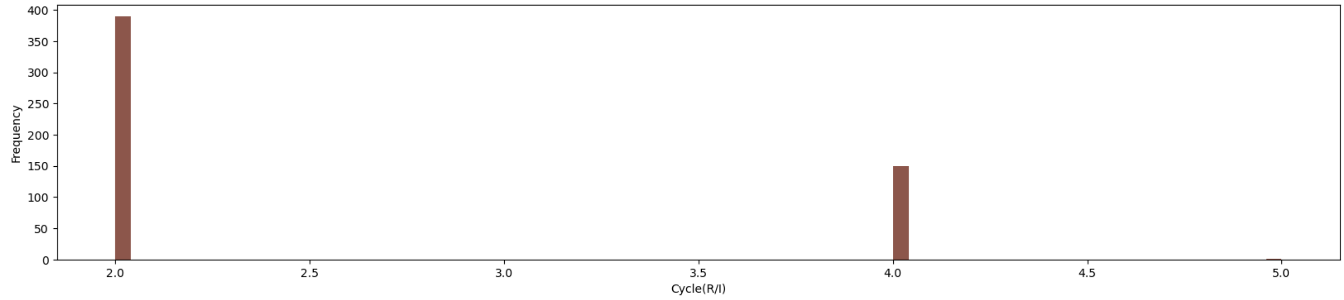

Frequency
0
10
20
30
40
50
60
70
140
150
160
170
180
Height(Cm)

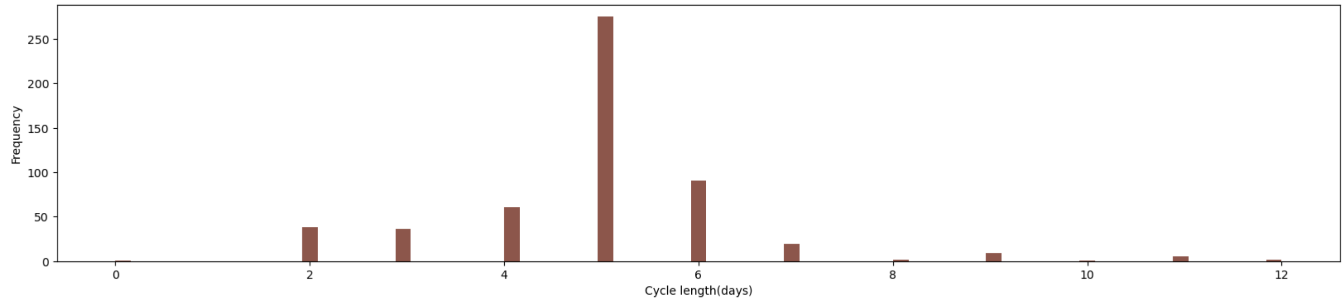

Frequency
0
50
100
150
200
250
300
15
20
25
30
35
40
BMI

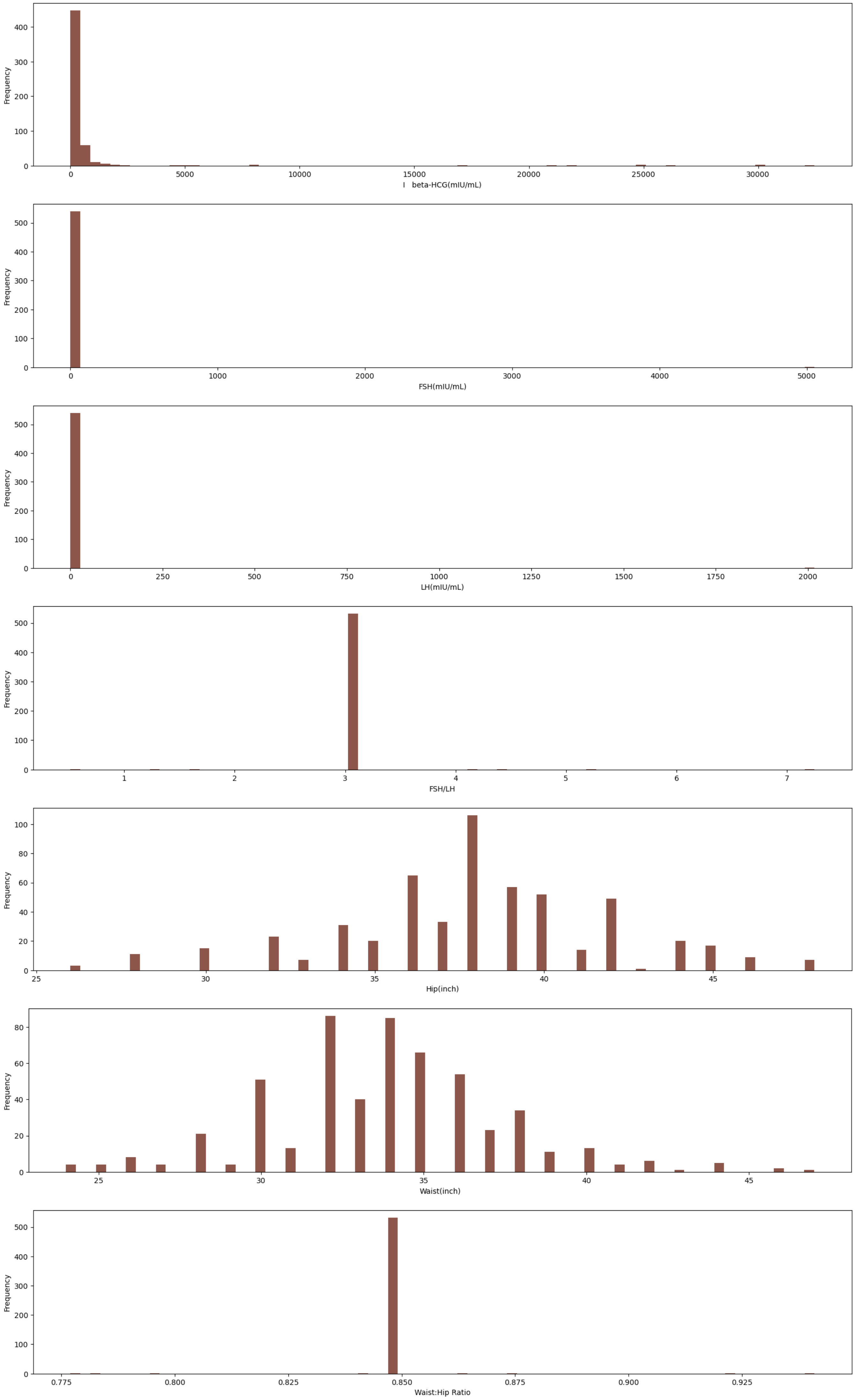

Frequency
0
50
100
150
200
250
10
20
30
40
50
60
70
80
Pulse rate(bpm)

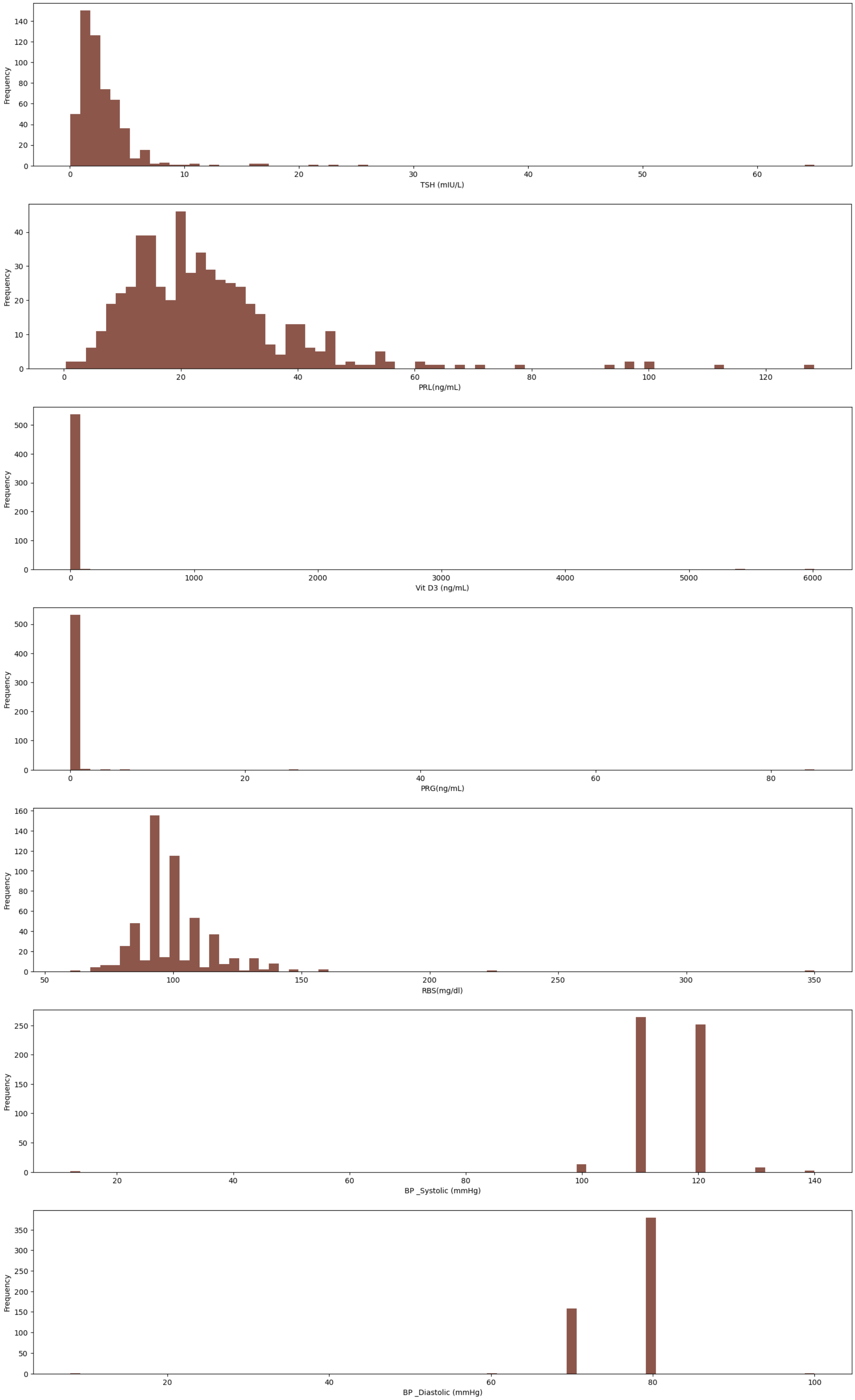

Frequency
0
50
100
150
200
250
16
18
20
22
24
26
28
RR (breaths/min)

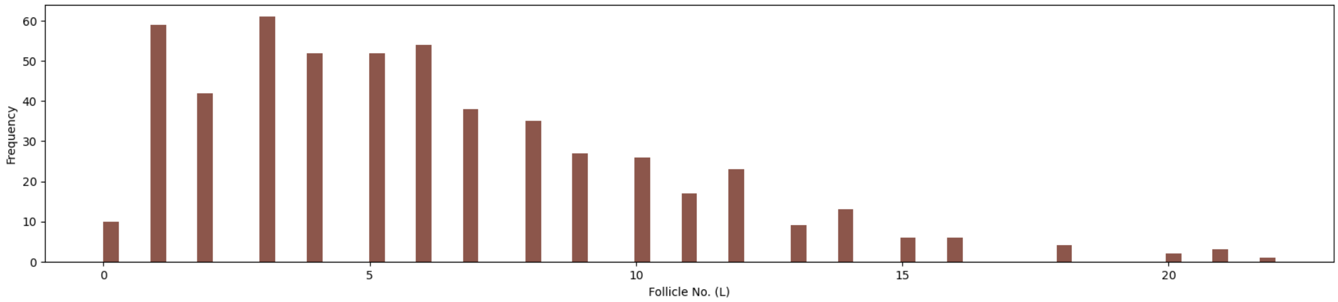

Frequency
0
10
20
30
40
50
60
9
10
11
12
13
14
15
Hb(g/dl)

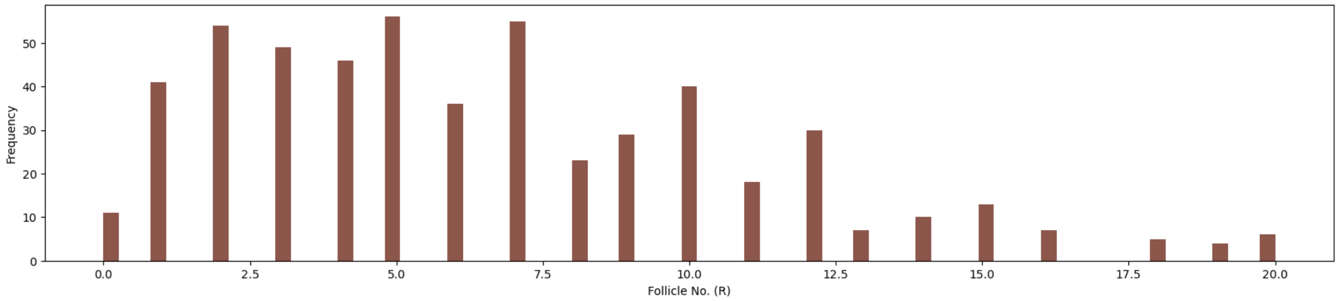

Frequency
0
50
100
150
200
250
300
350
400
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Cycle(R/I)

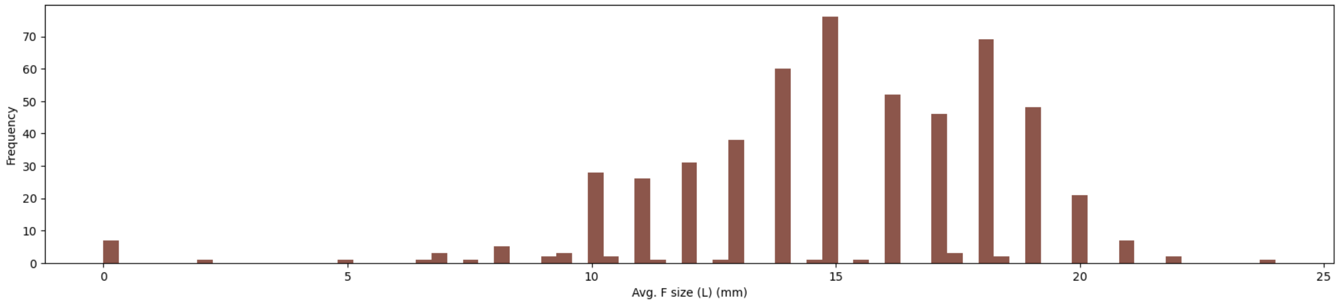

Frequency
0
50
100
150
200
250
0
2
4
6
8
10
12
Cycle length(days)

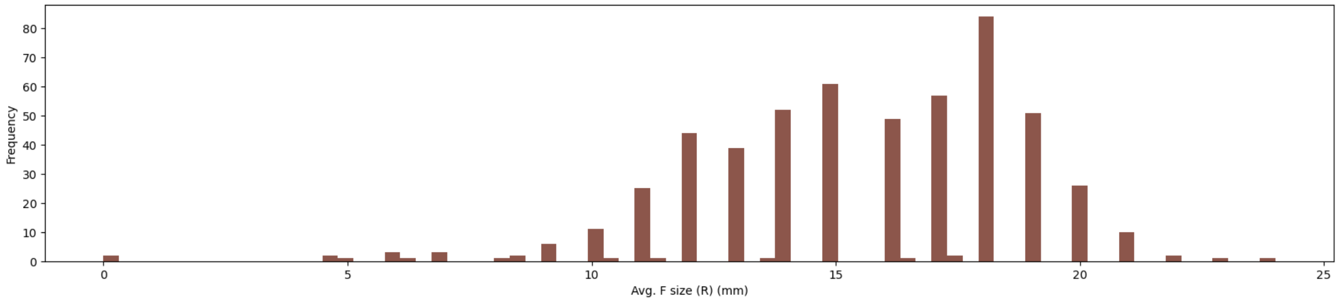

Frequency
400
300
200
100
0
0
5000
10000
15000
20000
25000
30000
I   beta-HCG(mIU/mL)
Frequency
500
400
300
200
100
0
0
1000
2000
3000
4000
5000
FSH(mIU/mL)
Frequency
500
400
300
200
100
0
0
250
500
750
1000
1250
1500
1750
2000
LH(mIU/mL)
Frequency
500
400
300
200
100
0
1
2
3
4
5
6
7
FSH/LH
Frequency
100
80
60
40
20
0
25
30
35
40
45
Hip(inch)
Frequency
80
60
40
20
0
25
30
35
40
45
Waist(inch)
Frequency
500
400
300
200
100
0
0.775
0.800
0.825
0.850
0.875
0.900
0.925
Waist:Hip Ratio

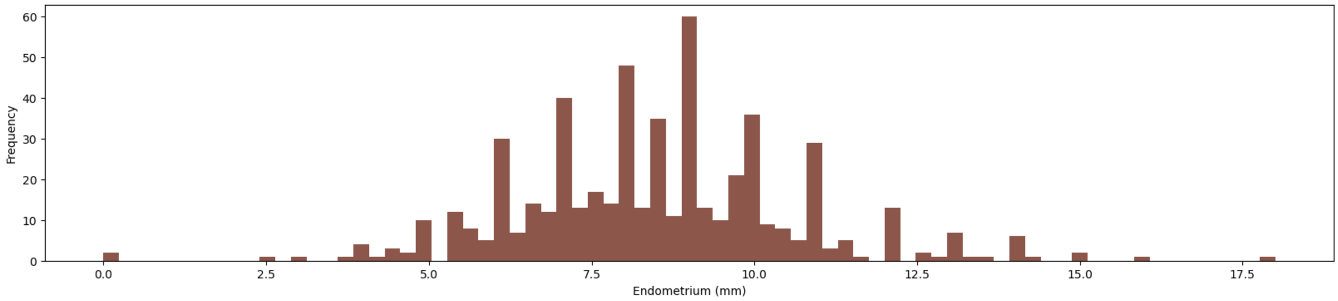

Frequency
0
20
40
60
80
100
120
140
0
10
20
30
40
50
60
TSH (mIU/L)
Frequency
0
10
20
30
40
0
20
40
60
80
100
120
PRL(ng/mL)
Frequency
0
100
200
300
400
500
0
1000
2000
3000
4000
5000
6000
Vit D3 (ng/mL)
Frequency
0
100
200
300
400
500
0
20
40
60
80
PRG(ng/mL)
Frequency
0
20
40
60
80
100
120
140
160
50
100
150
200
250
300
350
RBS(mg/dl)
Frequency
0
50
100
150
200
250
20
40
60
80
100
120
140
BP _Systolic (mmHg)
Frequency
0
50
100
150
200
250
300
350
20
40
60
80
100
BP _Diastolic (mmHg)











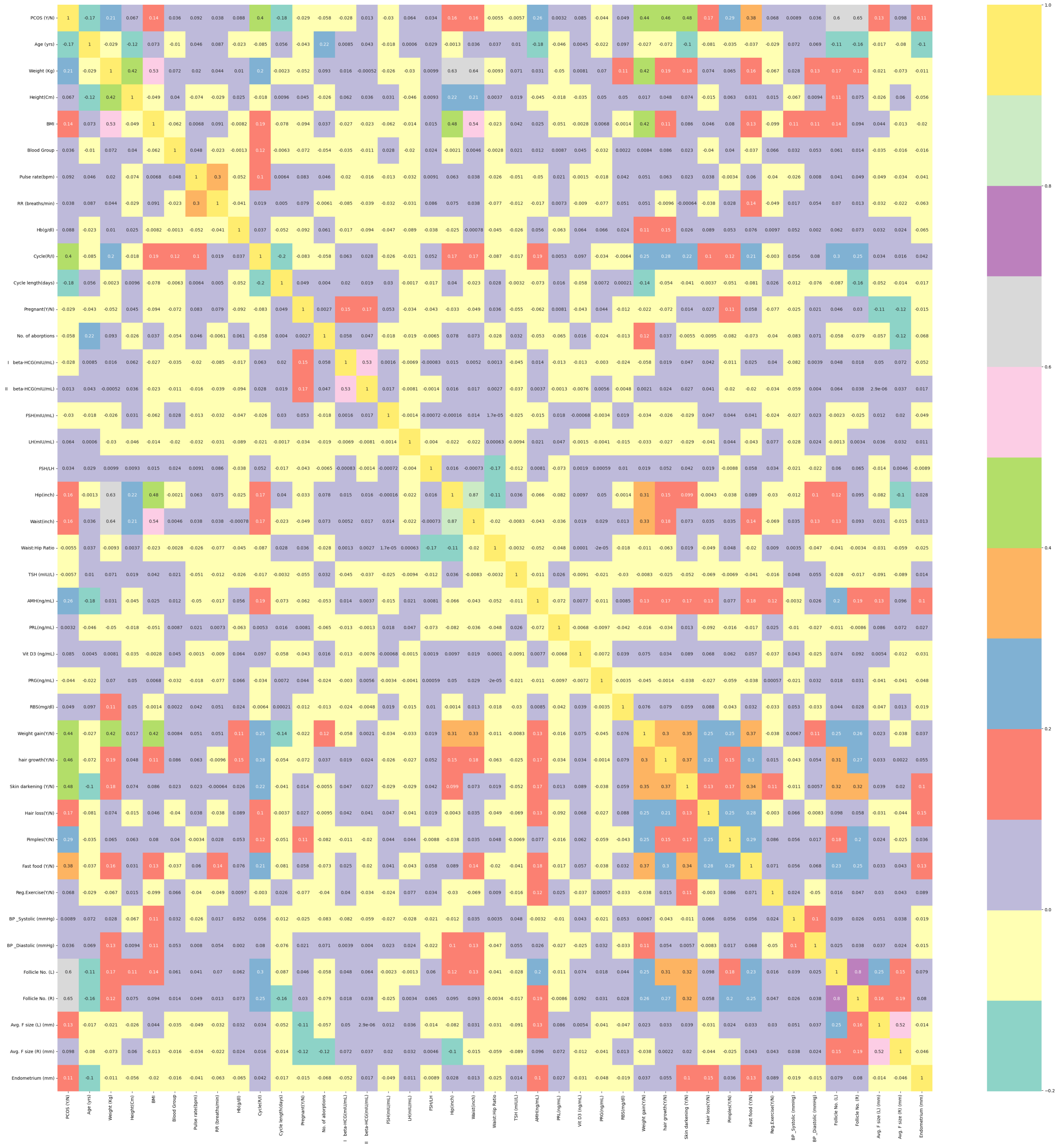
Feature Selection

Correlation can be positive (increase in one value of feature increases the value of the target variable) or negative (increase in one value of feature decreases the value of the target variable)

Using correlation with heatmap to identify the important features

In [439…

```
correlation_mat = pcos_normalized_dt.corr()

#get correlations of each features in dataset
feature_index = correlation_mat.index
plt.figure(figsize = (45,45))

#plot heat map
plot_heatmap = sns.heatmap(pcos_normalized_dt[feature_index].corr(),annot=True,cmap="Set3")
```



In [440…

```
correlation_mat['PCOS (Y/N)'].sort_values(ascending=False)
```

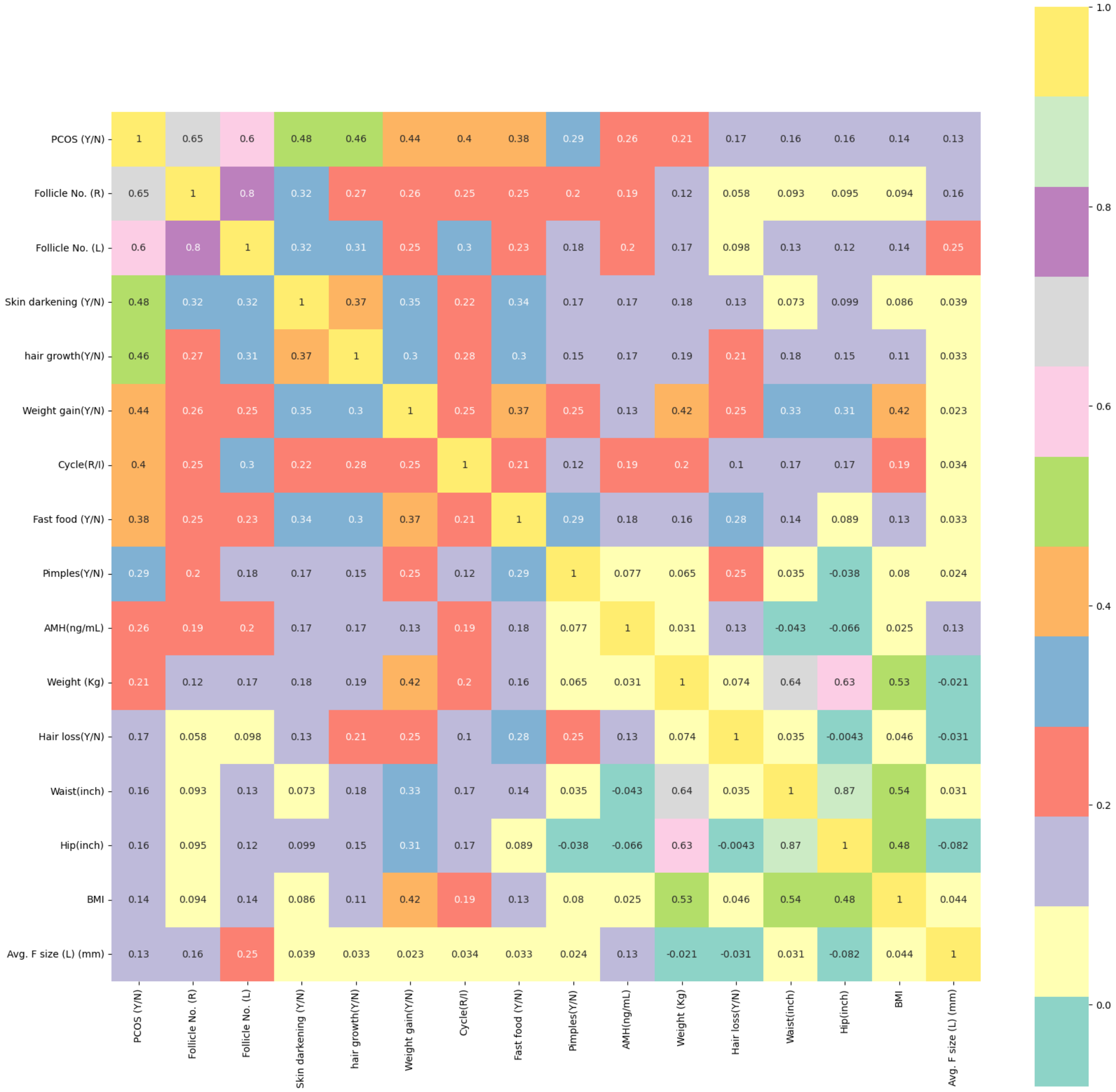
Out[440]:

```
PCOS (Y/N)                1.000000
Follicle No. (R)          0.648223
Follicle No. (L)          0.603109
Skin darkening (Y/N)      0.475283
hair growth(Y/N)          0.464245
Weight gain(Y/N)          0.440488
Cycle(R/I)                0.401165
Fast food (Y/N)           0.375389
Pimples(Y/N)              0.287802
AMH(ng/mL)                0.263863
Weight (Kg)               0.211628
Hair loss(Y/N)            0.171913
Waist(inch)               0.164378
Hip(inch)                 0.161480
BMI                       0.135256
Avg. F size (L) (mm)      0.133808
Endometrium (mm)          0.105151
Avg. F size (R) (mm)      0.097950
Pulse rate(bpm)           0.092084
Hb(g/dl)                  0.088046
Vit D3 (ng/mL)            0.085491
Reg.Exercise(Y/N)         0.067809
Height(Cm)                0.067358
LH(mIU/mL)                0.063817
RBS(mg/dl)                0.048956
RR (breaths/min)          0.037530
BP _Diastolic (mmHg)      0.036494
Blood Group               0.035892
FSH/LH                    0.033682
II    beta-HCG(mIU/mL)    0.012576
BP _Systolic (mmHg)       0.008885
PRL(ng/mL)                0.003243
Waist:Hip Ratio          -0.005525
TSH (mIU/L)              -0.005726
I   beta-HCG(mIU/mL)     -0.027870
Pregnant(Y/N)            -0.028606
FSH(mIU/mL)              -0.030403
PRG(ng/mL)               -0.043960
No. of aborptions        -0.057732
Age (yrs)                -0.167422
Cycle length(days)       -0.178509
Name: PCOS (Y/N), dtype: float64
```

Selecting the top 15 features with highest p-value

In [441…

```
imp_features = correlation_mat.nlargest(16,'PCOS (Y/N)')['PCOS (Y/N)'].index
plt.figure(figsize = (20,20))
plot_heat_map_after = sns.heatmap(pcos_normalized_dt[imp_features].corr(),annot=True,square=T
                                  cmap="Set3",annot_kws={'size':10})
```



In [442…

```
correlation_mat.nlargest(16,'PCOS (Y/N)')['PCOS (Y/N)']
```

Out[442]:

```
PCOS (Y/N)              1.000000
Follicle No. (R)        0.648223
Follicle No. (L)        0.603109
Skin darkening (Y/N)    0.475283
hair growth(Y/N)        0.464245
Weight gain(Y/N)        0.440488
Cycle(R/I)              0.401165
Fast food (Y/N)         0.375389
Pimples(Y/N)            0.287802
AMH(ng/mL)              0.263863
Weight (Kg)             0.211628
Hair loss(Y/N)          0.171913
Waist(inch)             0.164378
Hip(inch)               0.161480
BMI                     0.135256
Avg. F size (L) (mm)    0.133808
Name: PCOS (Y/N), dtype: float64
```

In [443…

```
imp_features
```

Out[443]:

```
Index(['PCOS (Y/N)', 'Follicle No. (R)', 'Follicle No. (L)',
       'Skin darkening (Y/N)', 'hair growth(Y/N)', 'Weight gain(Y/N)',
       'Cycle(R/I)', 'Fast food (Y/N)', 'Pimples(Y/N)', 'AMH(ng/mL)',
       'Weight (Kg)', 'Hair loss(Y/N)', 'Waist(inch)', 'Hip(inch)', 'BMI',
       'Avg. F size (L) (mm)'],
      dtype='object')
```

In [444…

```
pcos_df = pcos_normalized_dt[imp_features]
```

In [445…

```
pcos_df.head()
# pcos_df.shape
```

Out[445]:

	PCOS (Y/N)	Follicle No. (R)	Follicle No. (L)	Skin darkening (Y/N)	hair growth(Y/N)	Weight gain(Y/N)	Cycle(R/I)	Fast food (Y/N)	Pimples(Y/N)	AMH(ng/mL)
0	0.0	0.15	0.136364	0.0	0.0	0.0	0.0	1.0	0.0	0.029894
1	0.0	0.25	0.136364	0.0	0.0	0.0	0.0	0.0	0.0	0.021700
2	1.0	0.75	0.590909	0.0	0.0	0.0	0.0	1.0	1.0	0.099090
3	0.0	0.10	0.090909	0.0	0.0	0.0	0.0	0.0	0.0	0.016995
4	0.0	0.20	0.136364	0.0	0.0	0.0	0.0	0.0	0.0	0.032777

Model

Dataset splitting

Splitting dataset into training, validation, and test sets.

In [446…

```
X = pcos_df.iloc[:,1:].values
y = pcos_df.iloc[:,0].values
```

In [447…

```
print(X.shape)
print(y.shape)
```

```
(540, 15)
(540,)
```

In [448…

```
X_train,X_test,y_train,y_test = train_test_split(X, y,test_size=0.3,random_state=189)
X_val,X_test,y_val,y_test = train_test_split(X_test, y_test,test_size=0.3,random_state=189)
```

1. Decision Tree

In [449…

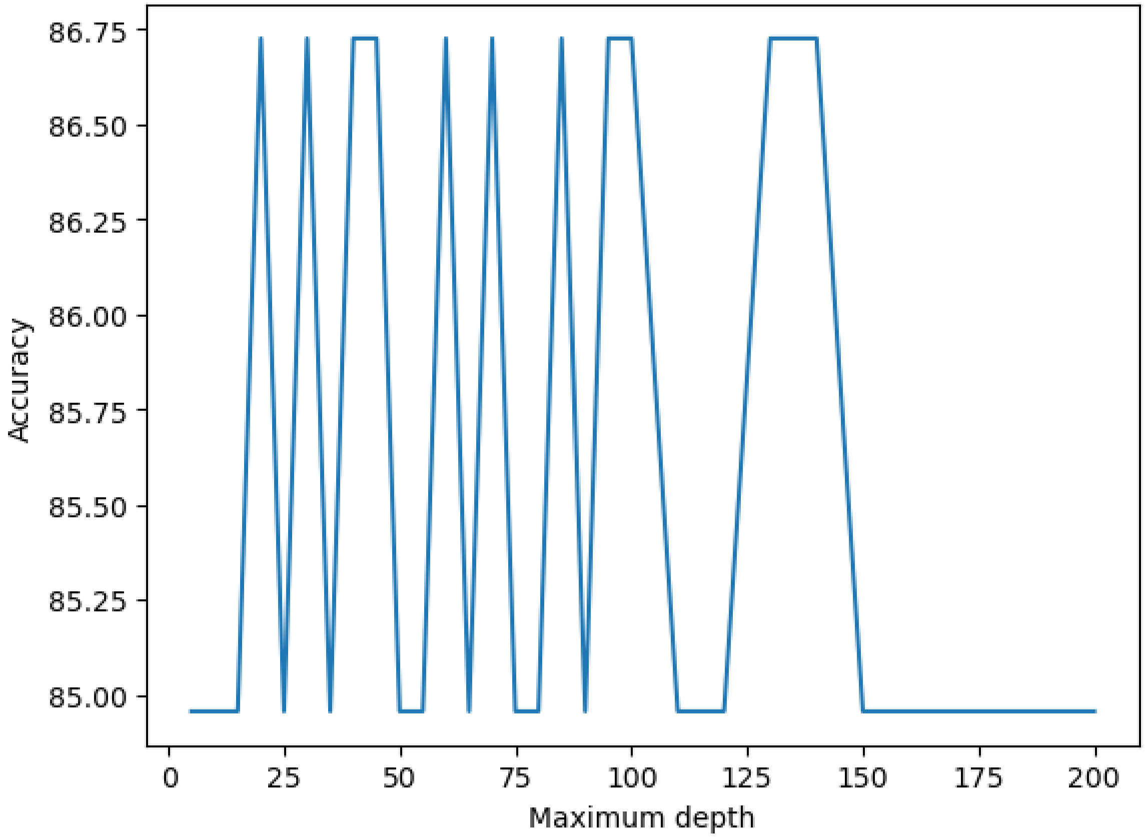
```
depths = [5,10,15,20,25,30,35,40,45,50,55,60,65,70,75,80,85,90,95,100,110,120,130,140,150,160
ac=[]
```

In [450…

```
for i in depths:
  dtree_clf = tree.DecisionTreeClassifier(max_depth=i,min_samples_leaf=4)
  dtree_clf.fit(X_train,y_train)
  y_pred_dtree = dtree_clf.predict(X_val)
  ac.append(accuracy_score(y_val,y_pred_dtree)*100)
```

In [451…

```
plt.plot(depths,ac,label='Test_accuracy')
plt.xlabel('Maximum depth')
plt.ylabel('Accuracy')
plt.show()
```



In [452…

```
#Finding the depth for which Accuracy is maximum

max_acc = max(ac)
max_dt = depths[ac.index(max_acc)]
print(max_acc, max_dt)
```

```
86.72566371681415 20
```

In [453…

```
#Using the depth which gave the maximum a15,ccuracy to train the model
dtree_clf = tree.DecisionTreeClassifier(max_depth=max_dt,min_samples_leaf=4)
dtree_clf.fit(X_train,y_train)
```

Out[453]:

```
▾                    DecisionTreeClassifier
DecisionTreeClassifier(max_depth=20, min_samples_leaf=4)
```

In [454…

```
y_pred_dtree = dtree_clf.predict(X_val)
```

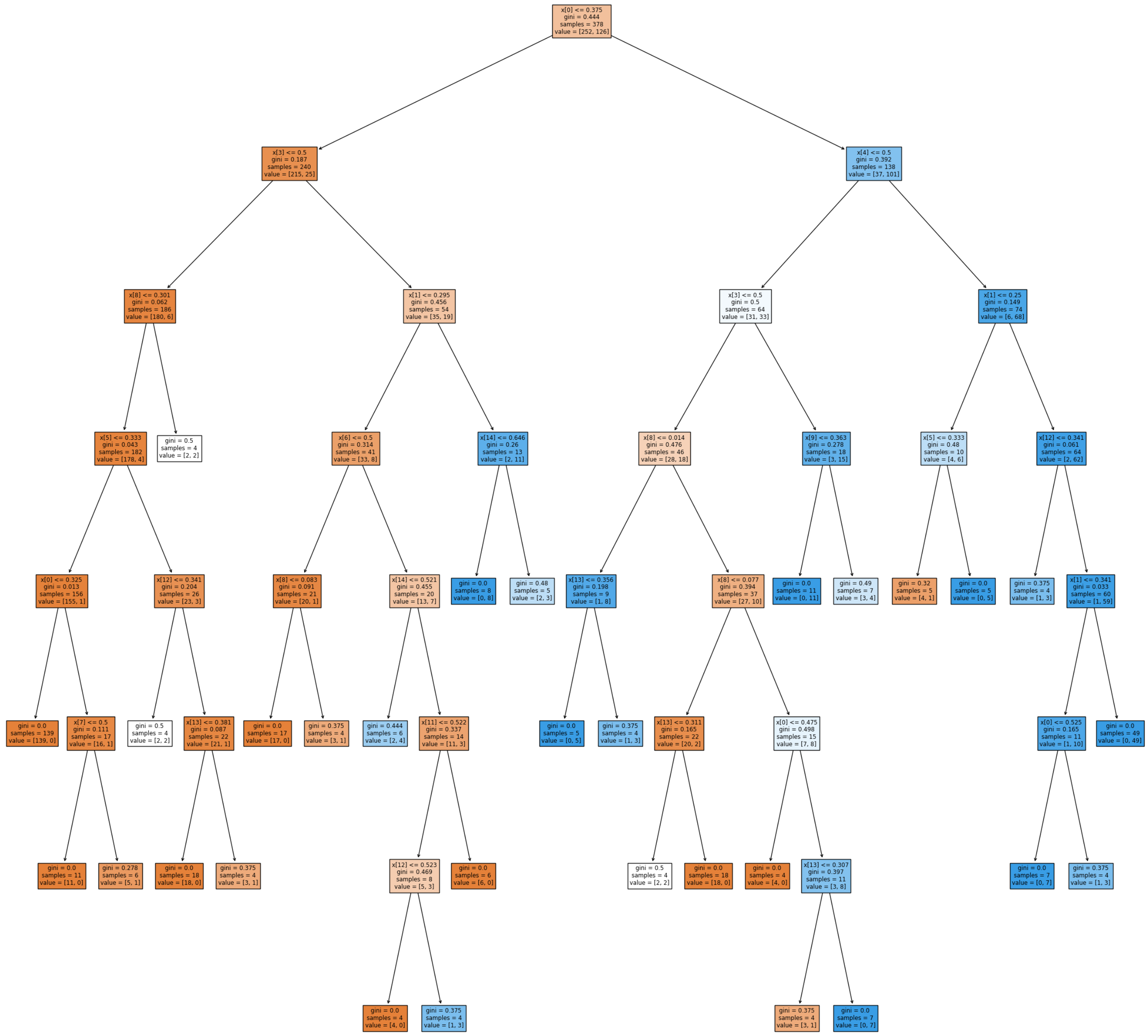
In [455…

```
acc_dtree = accuracy_score(y_val,y_pred_dtree)
print(acc_dtree)
```

```
0.8672566371681416
```

In [456…

```
fig = plt.figure(figsize=(30,30))
dtree_plot = tree.plot_tree(dtree_clf,filled=True)
```


x[0] <= 0.375 gini = 0.444 samples = 378 value = [252, 126]
x[3] <= 0.5 gini = 0.187 samples = 240 value = [215, 25]
x[4] <= 0.5 gini = 0.392 samples = 138 value = [37, 101]
x[8] <= 0.301 gini = 0.062 samples = 186 value = [180, 6]
x[1] <= 0.295 gini = 0.456 samples = 54 value = [35, 19]
x[3] <= 0.5 gini = 0.5 samples = 64 value = [31, 33]
x[1] <= 0.25 gini = 0.149 samples = 74 value = [6, 68]
x[5] <= 0.333 gini = 0.043 samples = 182 value = [178, 4]
gini = 0.5 samples = 4 value = [2, 2]
x[6] <= 0.5 gini = 0.314 samples = 41 value = [33, 8]
x[14] <= 0.646 gini = 0.26 samples = 13 value = [2, 11]
x[8] <= 0.014 gini = 0.476 samples = 46 value = [28, 18]
x[9] <= 0.363 gini = 0.278 samples = 18 value = [3, 15]
x[5] <= 0.333 gini = 0.48 samples = 10 value = [4, 6]
x[12] <= 0.341 gini = 0.061 samples = 64 value = [2, 62]
x[0] <= 0.325 gini = 0.013 samples = 156 value = [155, 1]
x[12] <= 0.341 gini = 0.204 samples = 26 value = [23, 3]
x[8] <= 0.083 gini = 0.091 samples = 21 value = [20, 1]
x[14] <= 0.521 gini = 0.455 samples = 20 value = [13, 7]
gini = 0.0 samples = 8 value = [0, 8]
gini = 0.48 samples = 5 value = [2, 3]
x[13] <= 0.356 gini = 0.198 samples = 9 value = [1, 8]
x[8] <= 0.077 gini = 0.394 samples = 37 value = [27, 10]
gini = 0.0 samples = 11 value = [0, 11]
gini = 0.49 samples = 7 value = [3, 4]
gini = 0.32 samples = 5 value = [4, 1]
gini = 0.0 samples = 5 value = [0, 5]
gini = 0.375 samples = 4 value = [1, 3]
x[1] <= 0.341 gini = 0.033 samples = 60 value = [1, 59]
gini = 0.0 samples = 139 value = [139, 0]
x[7] <= 0.5 gini = 0.111 samples = 17 value = [16, 1]
gini = 0.5 samples = 4 value = [2, 2]
x[13] <= 0.381 gini = 0.087 samples = 22 value = [21, 1]
gini = 0.0 samples = 17 value = [17, 0]
gini = 0.375 samples = 4 value = [3, 1]
gini = 0.444 samples = 6 value = [2, 4]
x[11] <= 0.522 gini = 0.337 samples = 14 value = [11, 3]
gini = 0.0 samples = 5 value = [0, 5]
gini = 0.375 samples = 4 value = [1, 3]
x[13] <= 0.311 gini = 0.165 samples = 22 value = [20, 2]
x[0] <= 0.475 gini = 0.498 samples = 15 value = [7, 8]
x[0] <= 0.525 gini = 0.165 samples = 11 value = [1, 10]
gini = 0.0 samples = 49 value = [0, 49]
gini = 0.0 samples = 11 value = [11, 0]
gini = 0.278 samples = 6 value = [5, 1]
gini = 0.0 samples = 18 value = [18, 0]
gini = 0.375 samples = 4 value = [3, 1]
x[12] <= 0.523 gini = 0.469 samples = 8 value = [5, 3]
gini = 0.0 samples = 6 value = [6, 0]
gini = 0.5 samples = 4 value = [2, 2]
gini = 0.0 samples = 18 value = [18, 0]
gini = 0.0 samples = 4 value = [4, 0]
x[13] <= 0.307 gini = 0.397 samples = 11 value = [3, 8]
gini = 0.0 samples = 7 value = [0, 7]
gini = 0.375 samples = 4 value = [1, 3]
gini = 0.0 samples = 4 value = [4, 0]
gini = 0.375 samples = 4 value = [1, 3]
gini = 0.375 samples = 4 value = [3, 1]
gini = 0.0 samples = 7 value = [0, 7]

Analyzation of the model

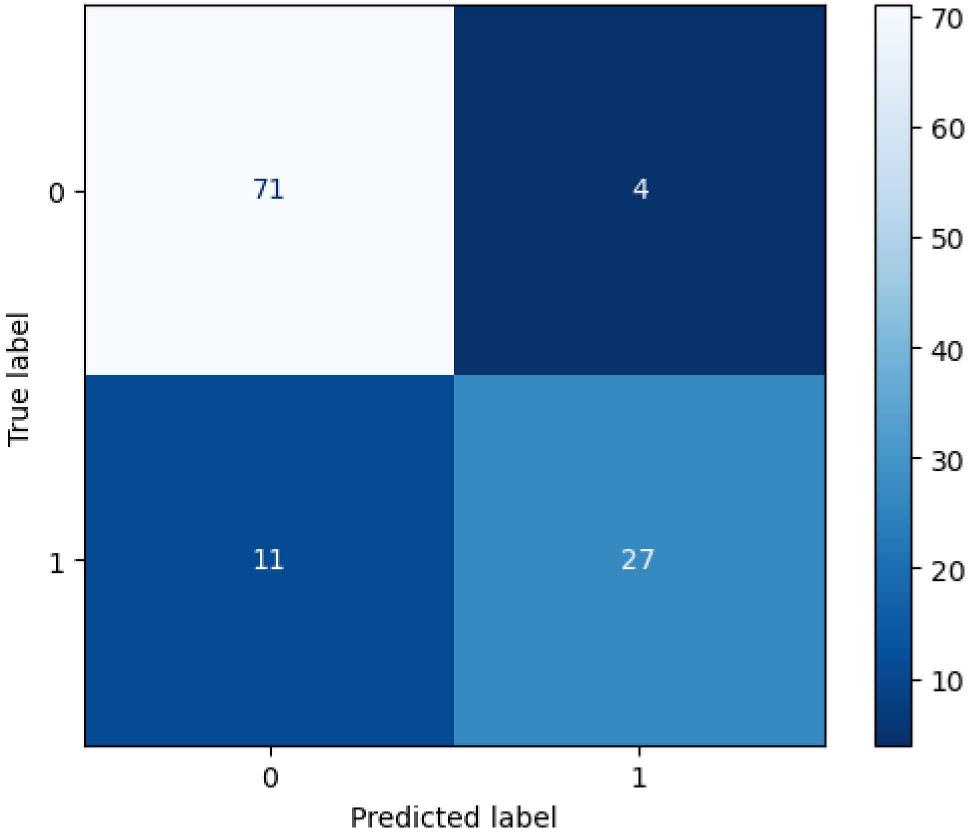
In [457…

```
print(metrics.classification_report(y_val, y_pred_dtree))
```

```
              precision    recall  f1-score   support

         0.0       0.87      0.95      0.90        75
         1.0       0.87      0.71      0.78        38

    accuracy                           0.87       113
   macro avg       0.87      0.83      0.84       113
weighted avg       0.87      0.87      0.86       113
```

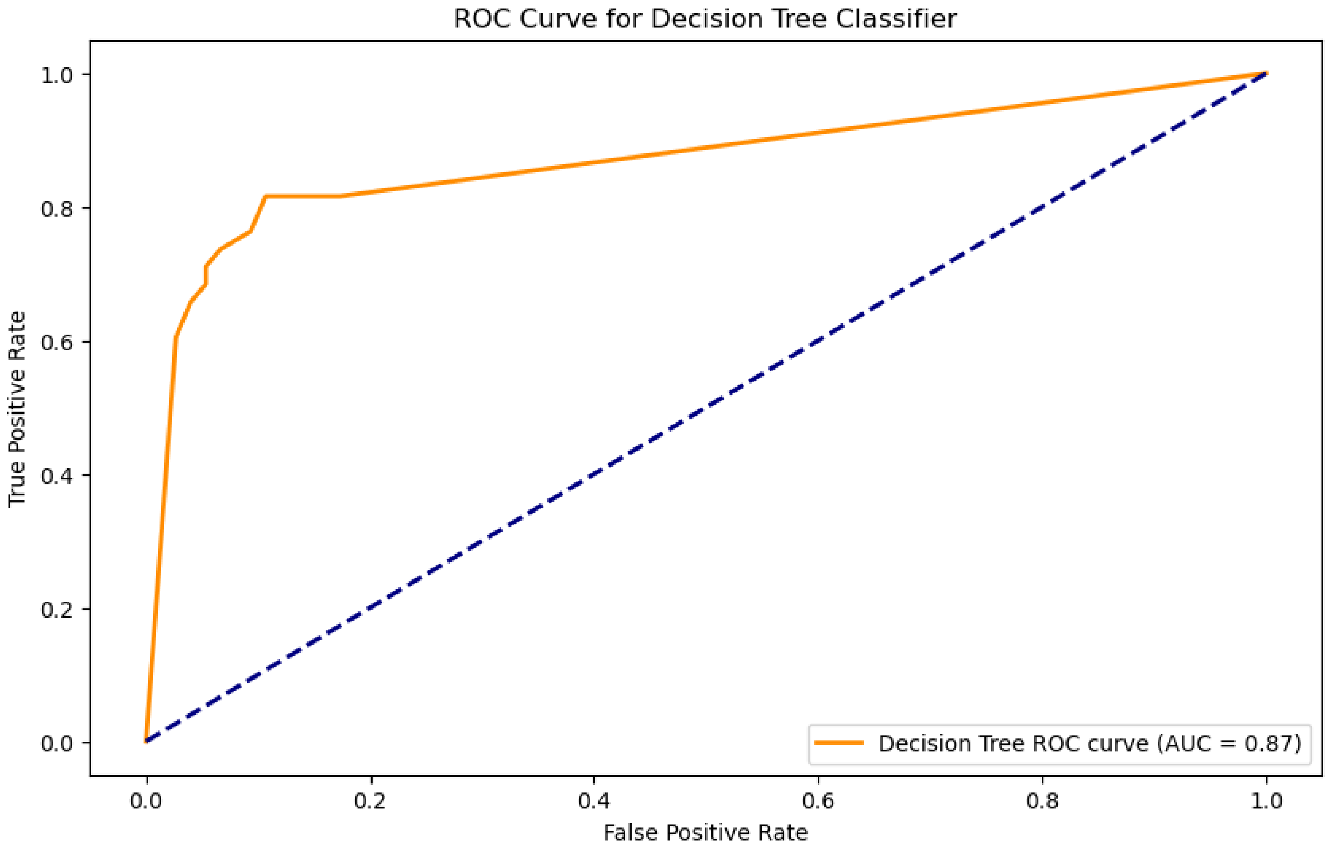
In [458…

```
cm_dree = metrics.confusion_matrix(y_val, y_pred_dtree)
disp = ConfusionMatrixDisplay(confusion_matrix = cm_dree, display_labels = ['0','1'])
disp.plot(cmap="Blues_r")
plt.show()
```



In [459…

```
from sklearn.metrics import roc_curve, auc
y_prob_dtree = dtree_clf.predict_proba(X_val)[:, 1]
fpr_dtree, tpr_dtree, thresholds_dtree = roc_curve(y_val, y_prob_dtree)
roc_auc_dtree = auc(fpr_dtree, tpr_dtree)
plot_roc_curve(fpr_dtree, tpr_dtree, roc_auc_dtree, 'Decision Tree')
```



In [460…

```
print(f'AUC for Decision Tree: {roc_auc_dtree:.4f}')
```

```
AUC for Decision Tree: 0.8705
```

2.SVM

In [461…

```
S = SVC(kernel = 'linear')
S.fit(X_train,y_train)
```

Out[461]:

```
▾ SVC
SVC(kernel='linear')
```

In [462…

```
y_pred_svm = S.predict(X_val)
```

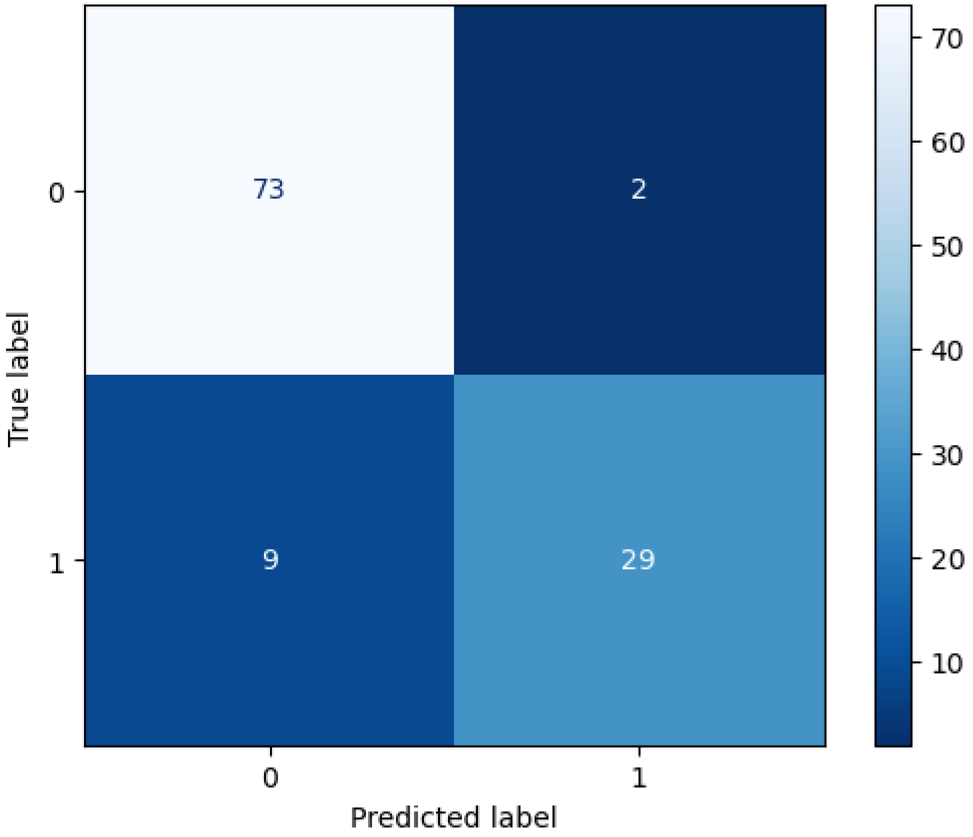
In [463…

```
acc_svm = accuracy_score(y_val,y_pred_svm)
print(acc_svm)
```

```
0.9026548672566371
```

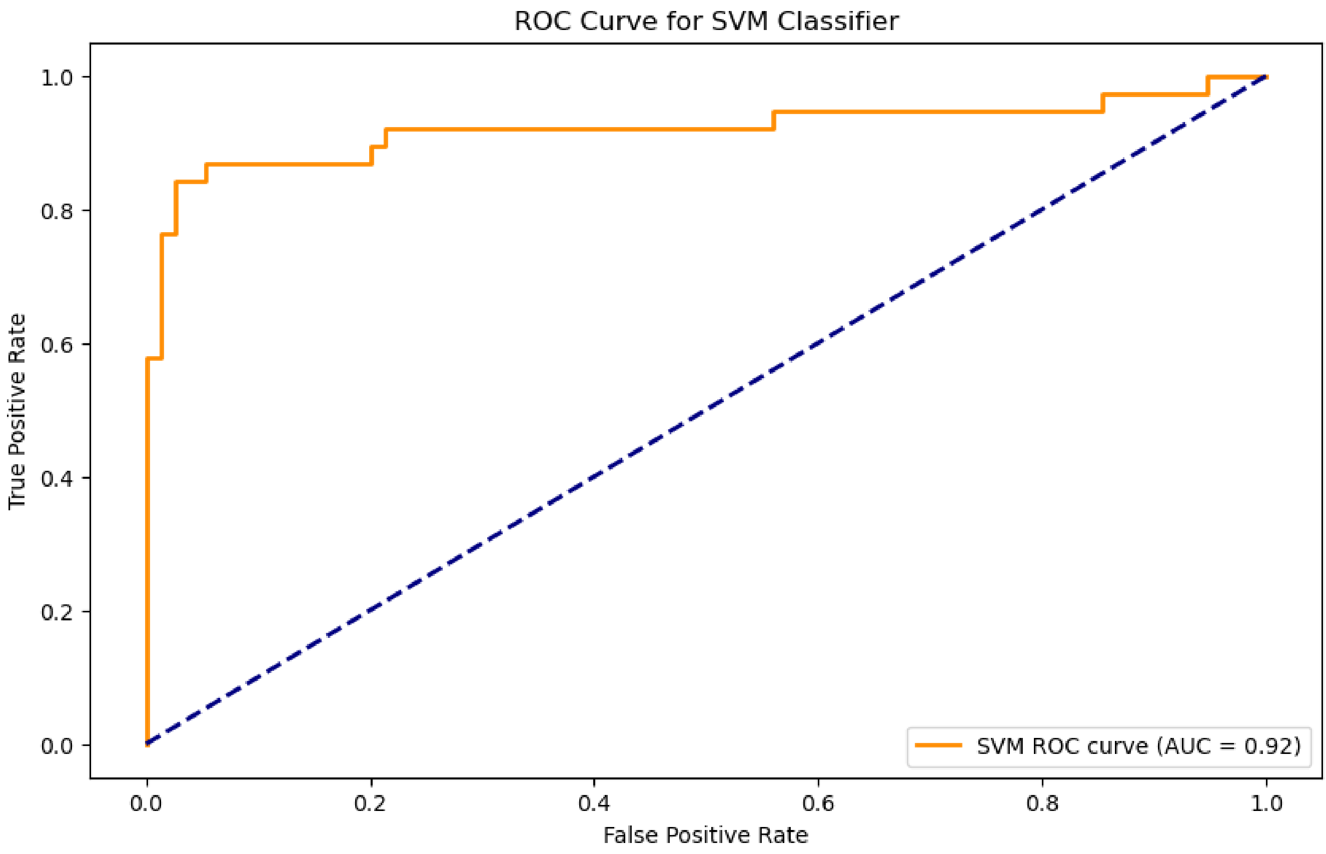
In [464…

```
cm_svm = metrics.confusion_matrix(y_val, y_pred_svm)
disp = ConfusionMatrixDisplay(confusion_matrix = cm_svm, display_labels = ['0','1'])
disp.plot(cmap="Blues_r")
plt.show()
```



In [465…

```
y_prob_svm = S.decision_function(X_val)  # decision_function for SVM instead of predict_proba
fpr_svm, tpr_svm, thresholds_svm = roc_curve(y_val, y_prob_svm)
roc_auc_svm = auc(fpr_svm, tpr_svm)
plot_roc_curve(fpr_svm, tpr_svm, roc_auc_svm, 'SVM')
```



In [466…

```
print(f'AUC for SVM: {roc_auc_svm:.4f}')
```

```
AUC for SVM: 0.9211
```

3.Naive Bayes classifier

In [467…

```
# Assuming X and y are your feature matrix and target variable
kf = KFold(n_splits=5, shuffle=True)

y_true_all = []  # List to store true values
y_pred_all = []  # List to store predicted values
accuracy_scores = []

for train_index, test_index in kf.split(X):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]

    # Gaussian Naive Bayes model
    gnb = GaussianNB()

    # Fit model with Laplace smoothing
    gnb.fit(X_train, y_train)

    # Make predictions
    y_pred = gnb.predict(X_test)

    # Save true and predicted values
    y_true_all = np.concatenate((y_true_all, y_test))
    y_pred_all = np.concatenate((y_pred_all, y_pred))

    # Evaluate cross-validation accuracy
    accuracy = accuracy_score(y_test, y_pred)
    accuracy_scores.append(accuracy)
```

In [468…

```
# Print average accuracy
print("Accuracy:", np.mean(accuracy_scores),"%")
```

```
Accuracy: 0.8814814814814813 %
```

In [469…

```
# Evaluate the accuracy
acc_nb = accuracy_score(y_test, y_pred)
print("Accuracy:", acc_nb * 100, "%")
```

```
Accuracy: 88.88888888888889 %
```

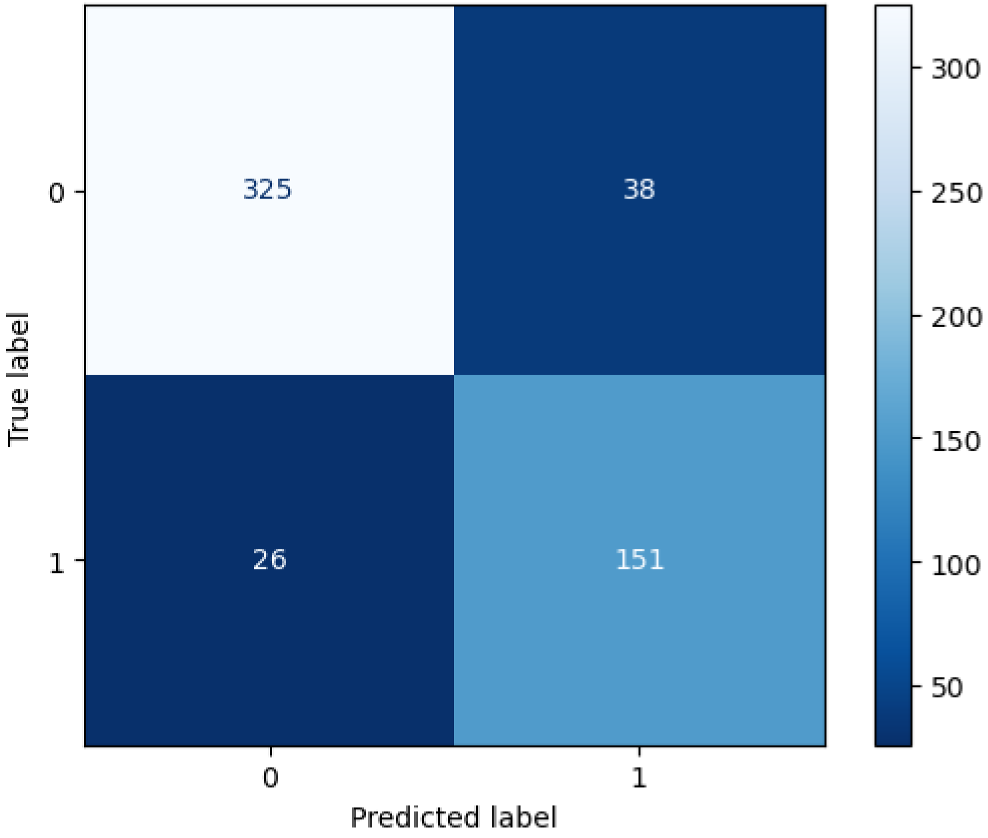
In [470…

```
# Evaluate overall performance
print("Overall Classification Report:")
print(classification_report(y_true_all, y_pred_all))
```

```
Overall Classification Report:
              precision    recall  f1-score   support

         0.0       0.93      0.90      0.91       363
         1.0       0.80      0.85      0.83       177

    accuracy                           0.88       540
   macro avg       0.86      0.87      0.87       540
weighted avg       0.88      0.88      0.88       540
```

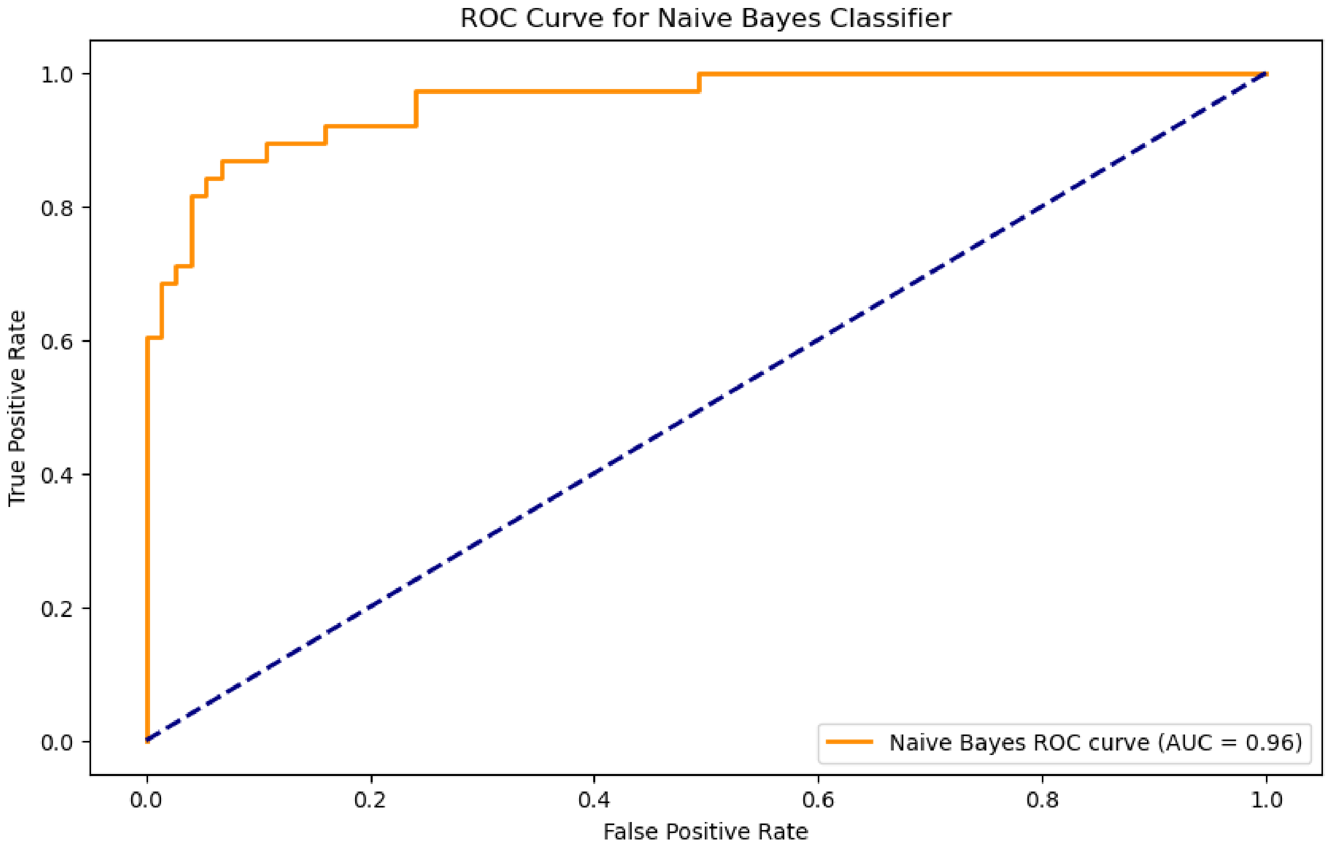
In [471…

```
# Assuming y_true_all and y_pred_all are your true and predicted labels
cm = confusion_matrix(y_true_all, y_pred_all)

# Plotting confusion matrix
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['0', '1'])
disp.plot(cmap="Blues_r")
plt.show()
```



In [472…

```
y_prob_nb = gnb.predict_proba(X_val)[:, 1]
fpr_nb, tpr_nb, thresholds_nb = roc_curve(y_val, y_prob_nb)
roc_auc_nb = auc(fpr_nb, tpr_nb)
plot_roc_curve(fpr_nb, tpr_nb, roc_auc_nb, 'Naive Bayes')
```



In [473…

```
print(f'AUC for Naive Bayes: {roc_auc_nb:.4f}')
```

```
AUC for Naive Bayes: 0.9582
```

4.XG Boost

In [474…

```
!pip install xgboost
```

```
Requirement already satisfied: xgboost in c:\users\vinay\anaconda3\lib\site-packages (2.0.3)
Requirement already satisfied: numpy in c:\users\vinay\anaconda3\lib\site-packages (from xgbo
ost) (1.24.3)
Requirement already satisfied: scipy in c:\users\vinay\anaconda3\lib\site-packages (from xgbo
ost) (1.11.1)
```

In [475…

```
# Train XGBoost model
model = XGBClassifier()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate predictions
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
Accuracy: 90.74%
```

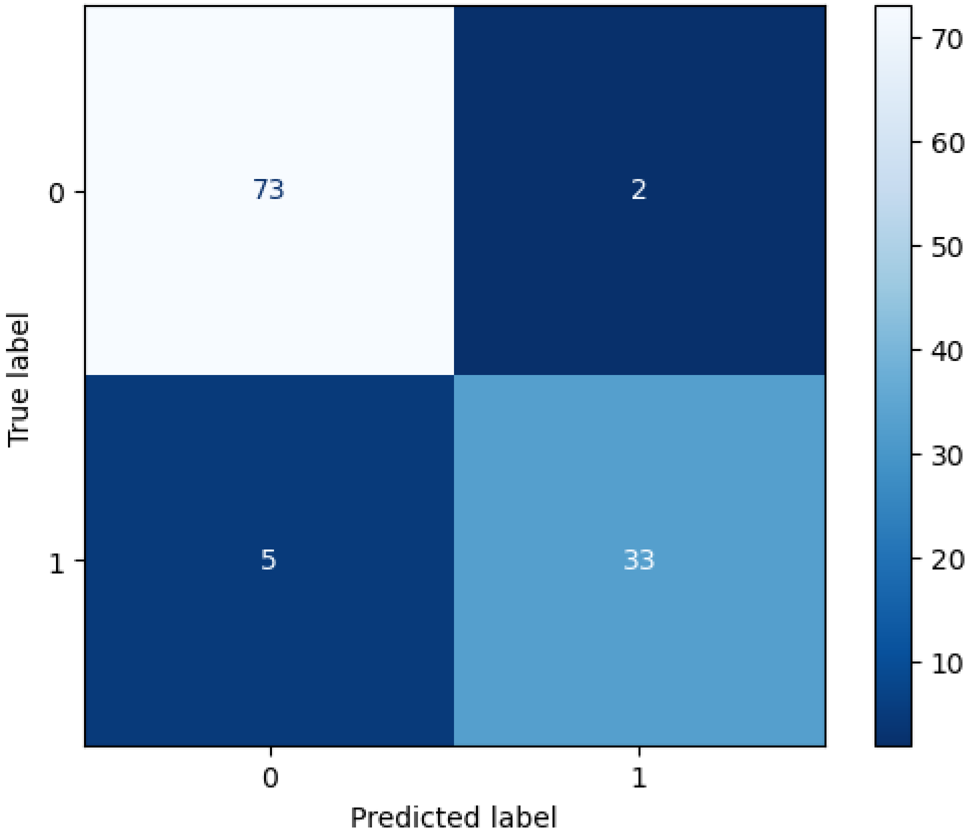
In [476…

```
y_pred_xgb = model.predict(X_val)

# Print the classification report
print(classification_report(y_val, y_pred_xgb))
```

```
              precision    recall  f1-score   support

         0.0       0.94      0.97      0.95        75
         1.0       0.94      0.87      0.90        38

    accuracy                           0.94       113
   macro avg       0.94      0.92      0.93       113
weighted avg       0.94      0.94      0.94       113
```

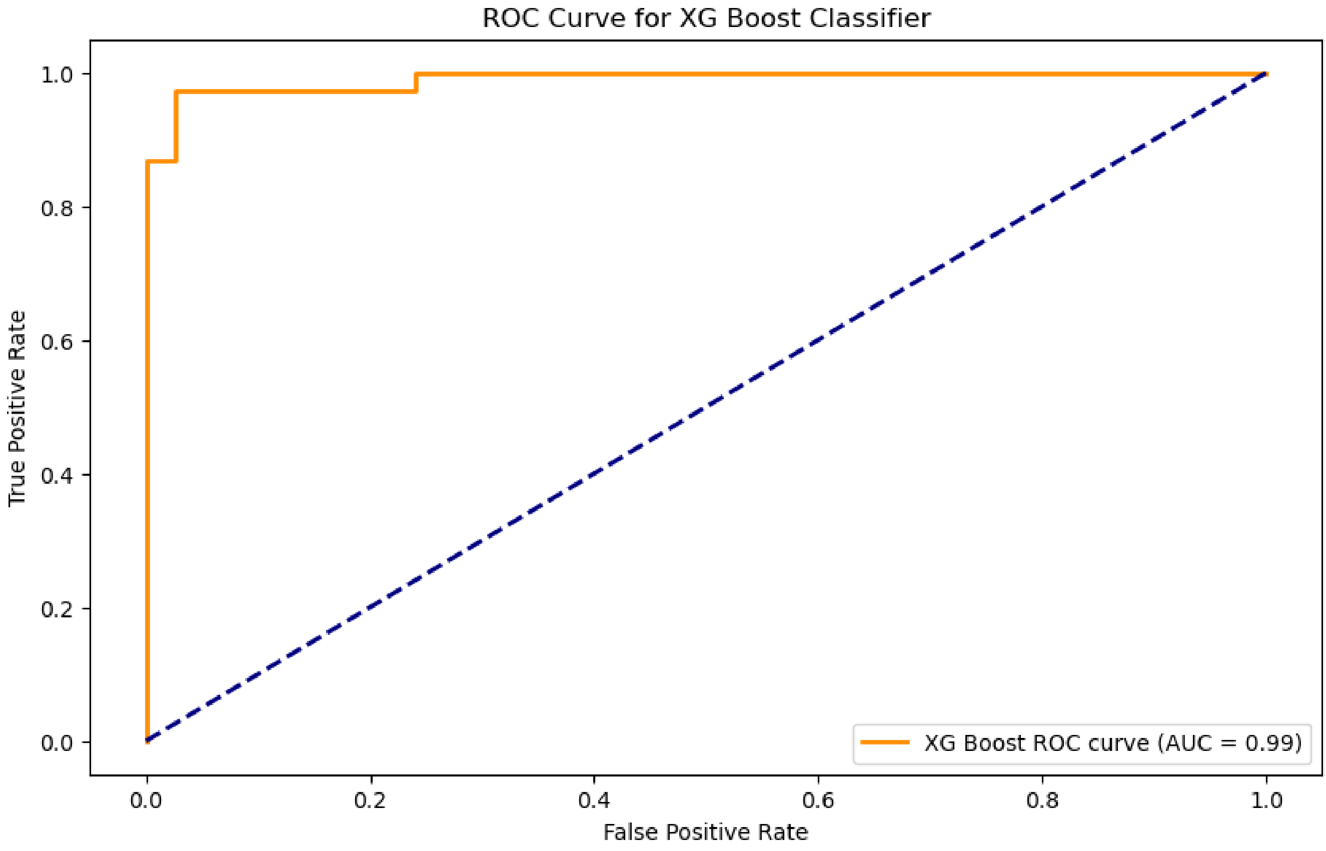
In [477…

```python
# Confusion Matrix
cm_xgb = confusion_matrix(y_val, y_pred_xgb)

# Plotting Confusion Matrix
disp_xgb = ConfusionMatrixDisplay(confusion_matrix=cm_xgb, display_labels=['0', '1'])
disp_xgb.plot(cmap="Blues_r")
plt.show()
```



In [478…

```python
y_prob_xgb = model.predict_proba(X_val)[:, 1]
fpr_xgb, tpr_xgb, thresholds_xgb = roc_curve(y_val, y_prob_xgb)
roc_auc_xgb = auc(fpr_xgb, tpr_xgb)
plot_roc_curve(fpr_xgb, tpr_xgb, roc_auc_xgb, 'XG Boost')
```



In [479…

```
print(f'AUC for XG Boost: {roc_auc_xgb:.4f}')
```

```
AUC for XG Boost: 0.9909
```

Comparing Different Models

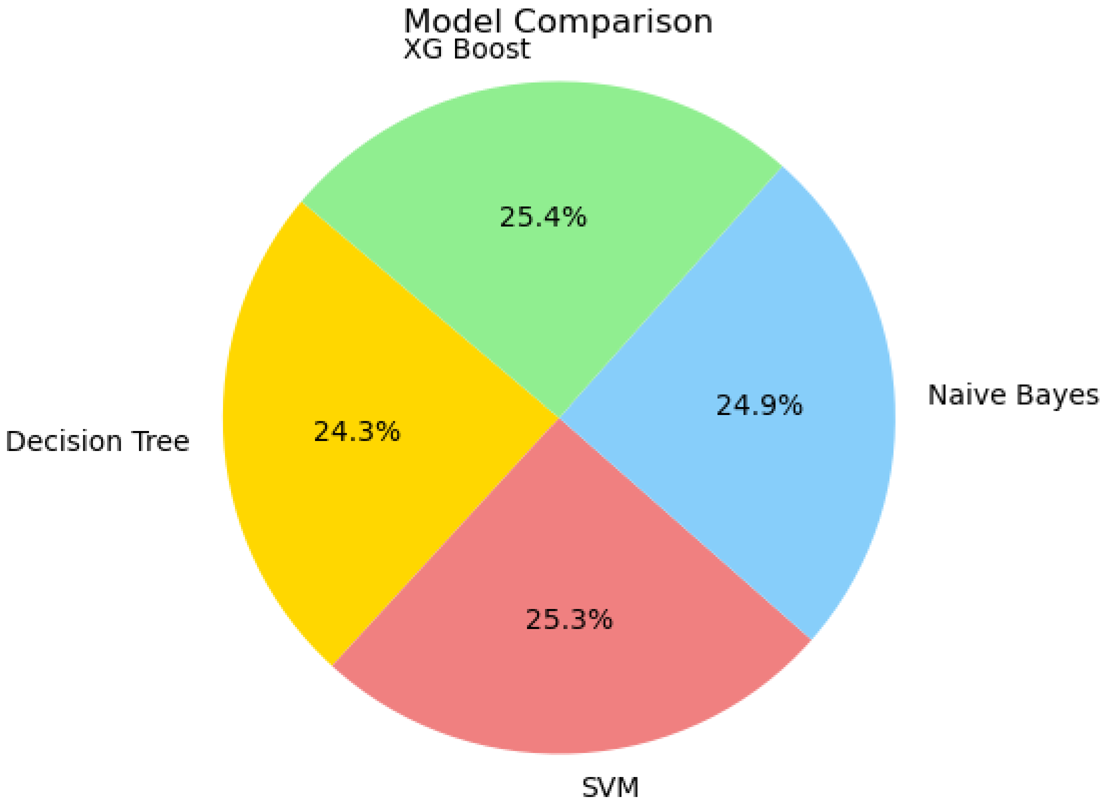
Using a box plot

In [480…

```
# Comparing Different Models - Pie Chart

# Create a dictionary to store the accuracies of different models
model_accuracies = {
    'Decision Tree': acc_dtree,
    'SVM': acc_svm,
    'Naive Bayes': acc_nb,
    'XG Boost': accuracy_score(y_test, y_pred)
}
```

In [481…

```
# Plotting the pie chart
labels = model_accuracies.keys()
sizes = model_accuracies.values()
colors = ['gold', 'lightcoral', 'lightskyblue', 'lightgreen']

plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangle=140)
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.title('Model Comparison')
plt.show()

plt.show()
```

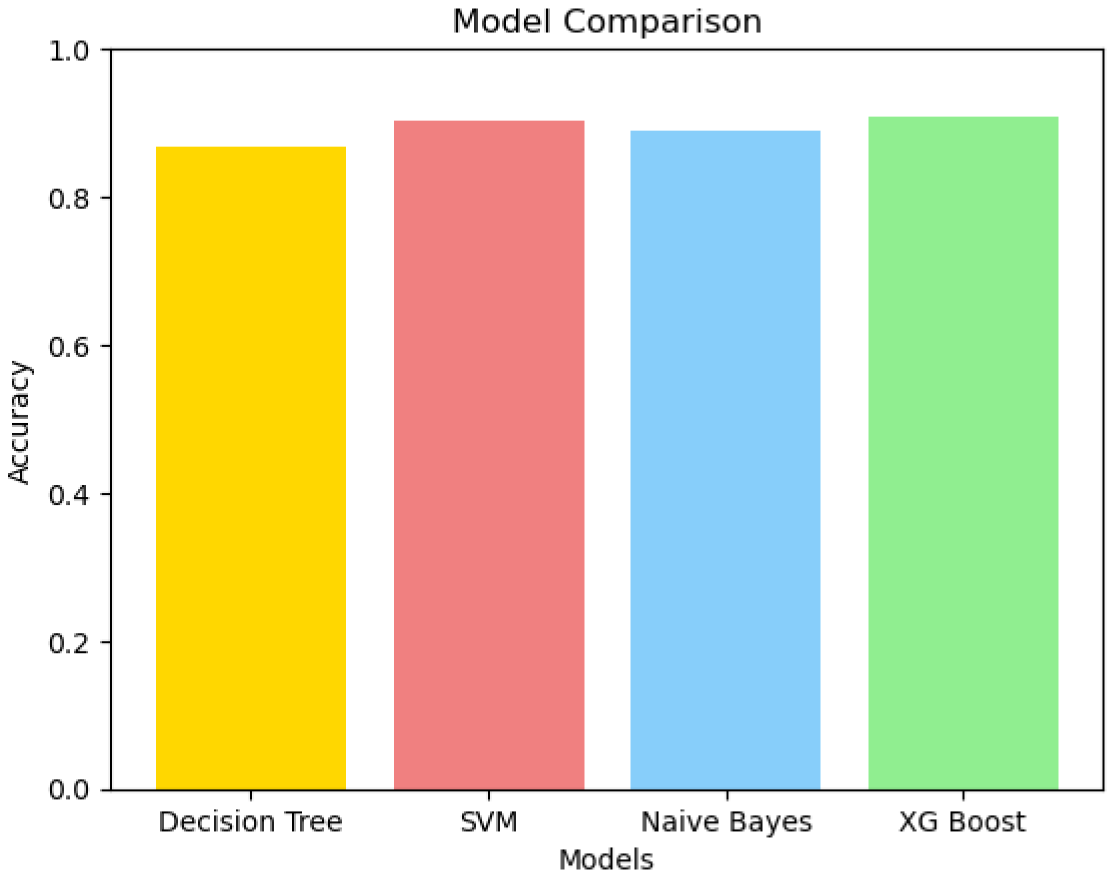


In [482…

```
# Comparing Different Models - Bar Graph

# Create a dictionary to store the accuracies of different models
model_accuracies = {
    'Decision Tree': acc_dtree,
    'SVM': acc_svm,
    'Naive Bayes': acc_nb,
    'XG Boost': accuracy_score(y_test, y_pred)
}
```

In [483…

```
# Plotting the bar graph
plt.bar(model_accuracies.keys(), model_accuracies.values(), color=['gold', 'lightcoral', 'lig
plt.xlabel('Models')
plt.ylabel('Accuracy')
plt.title('Model Comparison')
plt.ylim(0, 1)  # Set the y-axis limit between 0 and 1 for accuracy percentage
plt.show()
```



In [484…

```python
# Printing the accuracy results of each algorithm
print("Decision Tree Accuracy:", acc_dtree*100)
print("SVM Accuracy:", acc_svm*100)
print("Naive Bayes Accuracy:", acc_nb*100)
print("XG Boost Accuracy:", accuracy_score(y_test, y_pred)*100)
```

```
Decision Tree Accuracy: 86.72566371681415
SVM Accuracy: 90.2654867256637
Naive Bayes Accuracy: 88.88888888888889
XG Boost Accuracy: 90.74074074074075
```