Sakhi

Every Woman's Personal Health Assistant

Git Repo - https://github.com/raghavikonda22/SE_Project
Developed by - Group 4

Bhavana Nalabothu	BhavanaNalabothu@my.unt.edu
Charitha Sree Surineni	CharithaSreeSurineni@my.unt.edu
Srividya Kamakshi Valiveti	SrividyaKamakshiValiveti@my.unt.edu
Sai Sri Raghavi Konda	SaiSriRaghaviKonda@my.unt.edu
Hemalatha Yella	HemalathaYella@my.unt.edu

Women use period trackers for a variety of reasons, one main reason being the stigma around openly discussing menstruation and reproductive health. Although there is a substantial progress in creating awareness about menstruation, many women still feel uncomfortable mentioning these issues due to personal or culture reasons.

Sakhi is a comprehensive women's health application that aims to support and empower women at every stage of their health and reproductive journey. Sakhi is a Sanskrit word which means a female friend or companion. This name coveys a sense of support, trust, and companionship. In a health application which wants to provide personalized care to women, such qualities are. It helps women understand their bodies better so that they can make informed decisions about their reproductive health. With Sakhi, a user can:

- Track your menstrual cycle with ease and accuracy.
- Receive personalized insights on your fertile days and ovulation.
- Record and monitor your symptoms, moods, and overall health.
- Get access to information and resources on personal and reproductive health.
- Predict the risk of breast cancer, and other issues like PCOD/PCOS

1. Goals

The main aim of our project is to build a comprehensive reproductive health management tool. Following are some limitations of the existing solutions:

- *Lack of information and awareness about reproductive health:* Many women lack access to accurate information about their menstrual cycles, fertility, and reproductive health. Our tool will make helpful resources accessible to every woman. Based on this, users can act and get treatment if necessary. This increases their quality and longevity of life.

- *Limited support for women's health issues:* Many women face a variety of reproductive health issues such as PMS, PCOS, menopause, breast cancer which are not covered by the existing solutions in the market. Our application helps women to receive support for such health as well by tracking symptoms, risk assessment and constant monitoring.

- *Inadequate menstrual cycle tracking tools:* Many women struggle to keep track of their menstrual cycles due to irregularities or lack of convenient tracking tools. All solutions currently available in the market either require heavy in app purchases or their UI is covered with unwanted advertisements [2].

Sakhi incorporates a simple view of a calendar to plan and manage their periods effectively and free of cost. This calendar matches the user's mental model of a real calendar, so it becomes easy to use. Along with that, it has an option to predict a woman's risk of breast cancer based on personal and family history. Each user is prompted to give parameters like age, menstrual history, and family history as input. The application then calculates and predicts the user's risk of having breast cancer or PCOD/PCOS. After understanding the results, women can consider what steps should be taken next. If they want to receive some guidance, or take measures like medication etc., they can do so at the earliest.

1.1 Motivation

- Lot of women suffer from reproductive health issues like infertility, irregular periods, endometriosis, PCOS etc. According to the World Health Organization (WHO) [3], 10% of women around the world suffer from endometriosis and Polycystic ovary syndrome (PCOS) has been constant increasing with every year making it a global concern. Factors like diet, lifestyle and genetics influence a woman's reproductive health heavily. For example, being exposed to toxins in the environment (released due to polluted air, water etc.) or lack of exercise and consumption of processed foods. All these impact reproductive health negatively and create a hormonal imbalance.

- 1 in every 8 women are diagnosed with breast cancer and it's a leading cause of death [4]. These deaths can be prevented if the cancer is detected at the initial stage itself. For a woman to have this cancer, factors like age, history, genetic conditions play an important role. Similarly, PCOD/PCOS is an everyday problem for many. It causes irregular periods, unwanted hair growth along with other issues. Women have reduced chances of conceiving if diagnosed with this condition so treatment at the right time is important.

- Having applications like Sakhi can play an important role in empowering women to overcome these situations [1]. By addressing the prevalent reproductive health issues, we can significantly impact users and improve outcomes health-wise. Having modules like breast cancer prediction really helps women to ask for help at the right time. If the application predicts that a user is at high risk of breast cancer, she can get frequent scanning. If advised by the doctors, she can start taking medicines or undergo prophylactic surgery.

1.2 Significance

Following is a list that outlines what sets our initiative apart from others of a similar sort:

- *Wide range of health recommendations:* While the existing applications cover only some aspects of women's health, our tool covers a wide range of topics related to menstrual cycles, PCOD/PCOS, and cancer. This is why our application is a better choice for the users.
- Accurate recommendations: Sakhi has the potential to predict accurate dates of menstrual cycle and ovulation. This helps in learning more about their health and planning sexual engagements, vacations as per requirement. Sakhi also tracks blood flow during periods do that women can seek medical help whenever required. The ovulation dates that are predicted in the application play an important role for couples who are looking to conceive.
- *User-friendly interface:* The user interface of our application is simple and easy to use. Personalized health recommendations can be given to users as we keep track of their login credentials and settings. Therefore, it can reach a greater number of people.

- *Ease of management:* This website can track the menstrual flow, number of days in the cycle, ovulation days making it a one-stop solution for all reproductive health needs. It provides wholesome information on menstrual health making it stand out from the other applications.
- *Free of charge:* Most of the apps that concentrate on women's health require payment. Our application is completely free of cost. This helps even those women who don't have financial support to gain access to the application.
- *Addressing barriers to care:* Feeling uncomfortable to discuss menstrual health is quite common in women. If there's a platform that provides a safe space to manage these issues independently, it will help all these women to overcome the barriers and take the steps necessary to protect their reproductive health.

1.3 Objectives

Below are some objectives that have been identified for this project. Each objective is specific and measurable. This helps us as a team to focus only on those areas that are important, for development and ensure that the application includes all the relevant features.

- Create a user-friendly application that allows all women to add menstrual cycle data, receive insights about fertility and ovulation, and maintain their overall health.
- Test the application thoroughly by allowing users to give period data of various months as input. Allow them to add other aspects like symptoms, blood flow etc. over time.
- Train a machine learning model that will help in predicting the risk of a user to get breast cancer. This should be done based on personal data and family history.
- Give some resources in the application which can be used when users need guidance on various reproduction related topics.
- Identify all the areas which need technical improvement and work on the changes.

1.4 High-level Features

The Sakhi application is a user-friendly tool which can be used by anyone regardless of their technical expertise. Following are the features that the application will have:

- *User Authentication:* Users have the option to log in to the application. The application will store their personal information, such as name, age, and contact details followed by health data.
- *Period Tracking:* Users can track their menstrual cycles, by adding the start and end dates of each cycle, period duration of each month, and the intensity of blood flow.
- *Ovulation Prediction:* Menstrual cycle data is used to predict the most fertile days of the month. This shows users the best days for conceiving.
- *Breast Cancer Risk Detection:* The application collects information on the user's height, weight, family history, and other factors to predict their risk of developing breast cancer using a machine learning model [5].

- *Alerts and Notifications:* Alerts remind users of important dates, like the start of next period or the date of their next breast self-exam. Users can customize the frequency and notifications for a personalized experience.
- *Symptom monitoring:* Users can note down the symptoms they experienced during a period cycle like body pains, mood swings, discharges etc. like a journal. These can later be shown to a healthcare professional when needed.

2. Background Work

Sakhi is a women's health care application which has been built to provide health related help and support to women. The goal of this application is to bring all women reproductive health issues under one umbrella and bridge the gap between women's health care and education. In developing countries, many people are still deprived of quality health care and information. And applications like these are really needed there. Research shows that women go through various phases like menstruation, menopause, reproduction etc. but do not discuss them with anyon. If the resources are limited, and infrastructure is not so good, catering to the needs of each and every woman becomes challenging. This leads to delayed diagnosis, inadequate treatment, and poor health in women.

1. (Mehrotra et al., 2015) showed the importance of using internet in the health needs of women in areas with less resources.
2. Another study published in BMC Medical Informatics and Decision Making found that if a person uses a mobile app that gives information on menstrual health and contraception, their knowledge of contraception increases. Thereby, users use more contraception.
3. Journal of Obstetrics and Gynecology Canada found that if we keep track of the menstrual cycles, it will help in detecting and diagnosing reproductive issues like polycystic ovary syndrome (PCOS).
4. Journal of Medical Internet Research mentioned in a research that using menstrual tracking apps can improve women's knowledge and awareness of their menstrual cycles. This helps in detecting patterns in their cycles.
5. Journal of Clinical Oncology found that a machine learning model was able to accurately predict breast cancer risk in women, using data from factors like age, family history, and breast density.

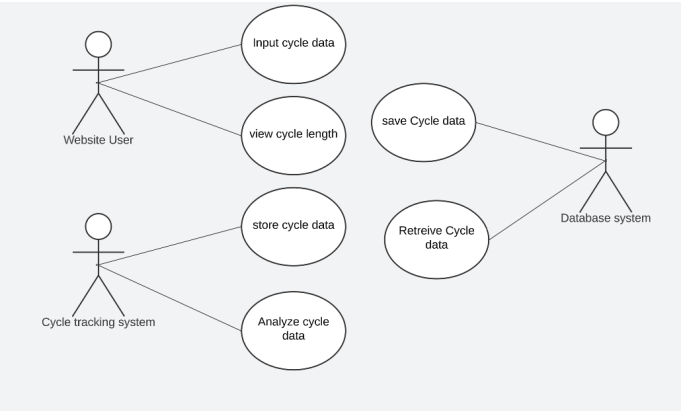
Overall, going through all these research articles makes us understand that predicting menstrual tracking and breast cancer can both be valuable in women's health care, and that machine learning models can be effective in predicting and managing these conditions.

However, there is still a need for more research on how to use technology appropriately so that each woman can be particularly catered to. The development of this application is a result of extensive research on women's health needs and the challenges they face these days. Sakhi has many features related to menstrual cycle tracking, pregnancy, breast cancer etc. and tries to provide personal support to the users.

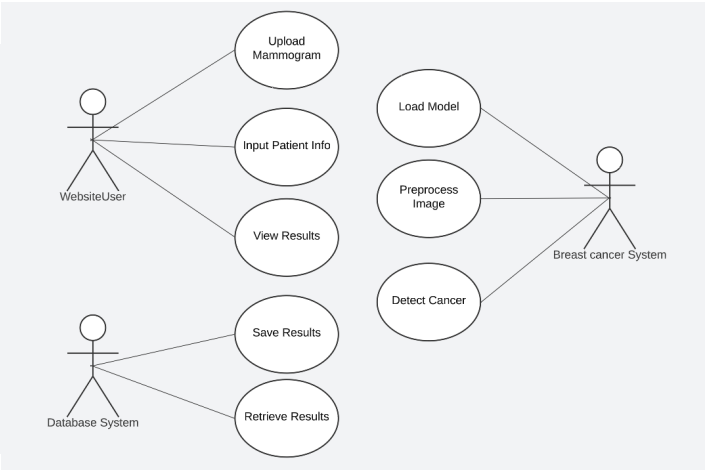
3. Increment I Features

3.1 UML Diagrams

3.1.1 Usecase Diagrams

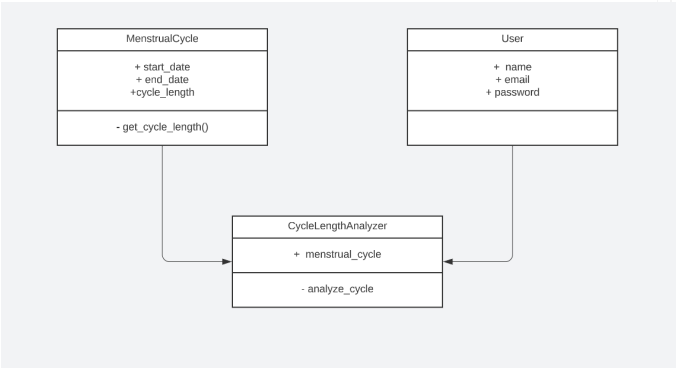


Menstrual Tracking

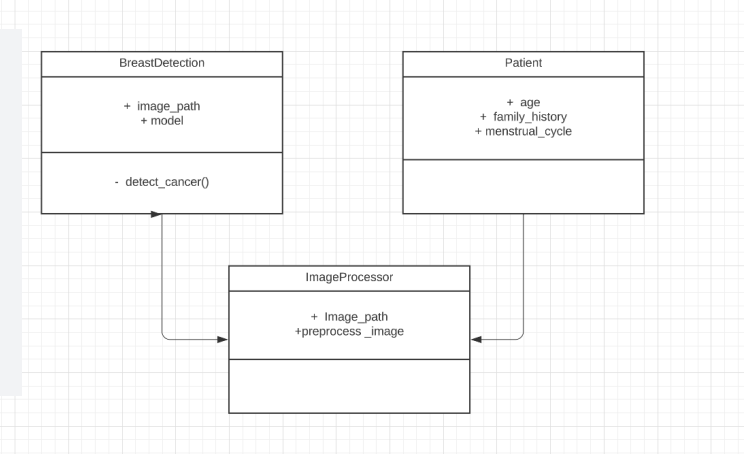


Breast Cancer

3.1.2 Class Diagrams



Menstrual Tracking



Breast Cancer

3.2 Breast Cancer Prediction

The machine learning model to predict breast cancer was developed using the logistic regression [10]. This output of the code is to diagnose whether the sample is cancerous or non-cancerous. The output would be predicted using the label of 0 and 1.

```
In [1]:
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

pd.options.display.max_columns = 100
```

Fig. 3.1

Fig. 3.1 is the first step in running the breast cancer prediction project. This step includes installing the required libraries that *numpy* and *pandas* [11]. These two are the most popular libraries used for the analysis.

```
In [ ]:
df = pd.read_csv("/content/data_breast_cancer.csv")
```

Fig. 3.2

Fig. 3.2 reads the data from *data_breast_cancer.csv* and creates the data frame object.

```
In [ ]:
df.head()
```

Out[19]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	...	concavity_worst	concave points_worst	symmetry_worst	fractal_dimension_worst	Unname
0	842302	M	17.99	10.38	122.80	...	0.7119	0.2654	0.4601	0.11890	Na
1	842517	M	20.57	17.77	132.90	...	0.2416	0.1860	0.2750	0.08902	Na
2	84300903	M	19.69	21.25	130.00	...	0.4504	0.2430	0.3613	0.08758	Na
3	84348301	M	11.42	20.38	77.58	...	0.6869	0.2575	0.6638	0.17300	Na
4	84358402	M	20.29	14.34	135.10	...	0.4000	0.1625	0.2364	0.07678	Na

5 rows × 33 columns

Fig. 3.3

3.2.1 Analysis

The next step here is df.head() This is used to display the first few rows of the data frame object that is just created above. Generally, It displays the first 5 rows in the data frame. (Fig. 3.3)

```
In [ ]:
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   id                       569 non-null    int64
 1   diagnosis                569 non-null    object
 2   radius_mean              569 non-null    float64
 3   texture_mean             569 non-null    float64
 4   perimeter_mean           569 non-null    float64
 5   area_mean                569 non-null    float64
 6   smoothness_mean          569 non-null    float64
 7   compactness_mean         569 non-null    float64
 8   concavity_mean           569 non-null    float64
 9   concave points_mean      569 non-null    float64
 10  symmetry_mean            569 non-null    float64
 11  fractal_dimension_mean   569 non-null    float64
 12  radius_se                569 non-null    float64
 13  texture_se               569 non-null    float64
```

Fig. 3.4

Fig. 3.4 shows technical information about the data frame. This step is generally performed to understand the data format, to find the missing null values, and to check whether the appropriate data type is assigned to each column or not.

3.2.2 Data visualization

```
In [ ]:
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go

%matplotlib inline
sns.set_style('darkgrid')
```

Fig. 3.5

Install all the necessary libraries that are used to create various charts and graphs (Fig. 3.5). It also includes installing famous libraries in python such as *matplot*, seaborn etc. *%matplotlib inline* allows the *jupyter notebook* to visualize the *matplotlib* plots inline.

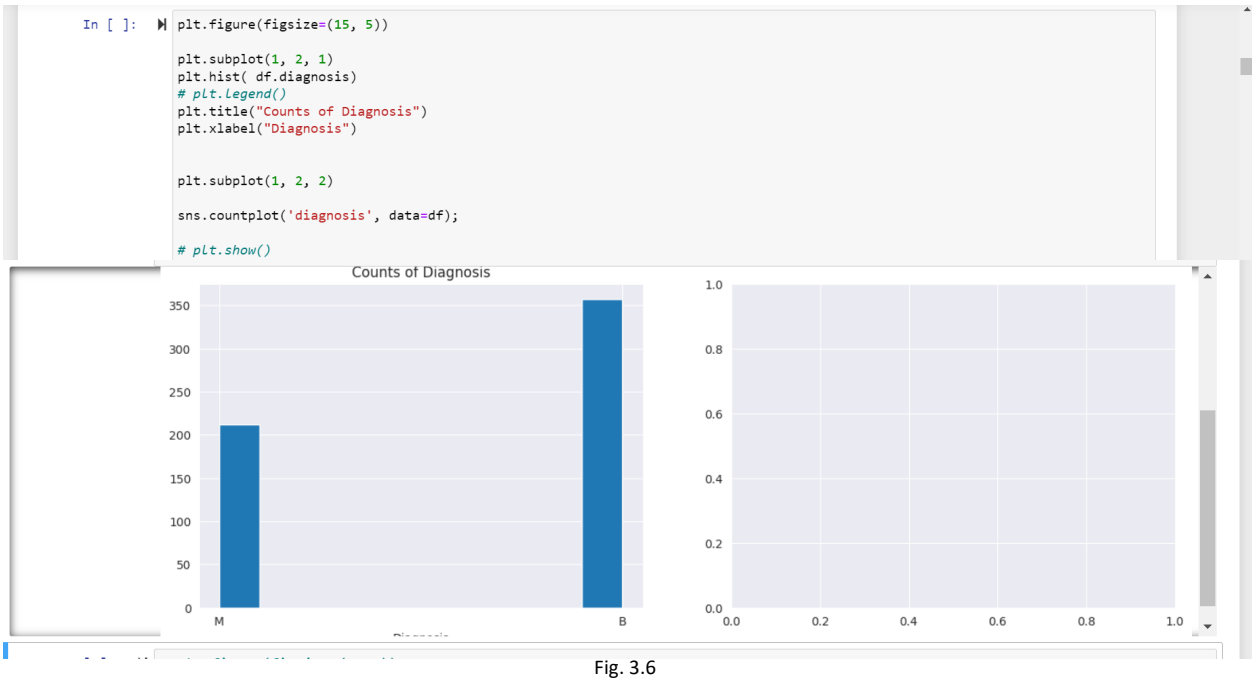


Fig. 3.6

Fig. 3.6 provides two types of plots for the column diagnosis column. It generates a *histogram* and *countplot*. This step is performed generally to understand the characteristics of the data and make decisions accordingly. This is also performed to identify the trends in the data and develop the model and make decisions accordingly.

Following are the statements to visualize the diagnosis column in the data frame object using the interactive histogram model.

```
In [ ]:
# plt.figure(figsize=(7,12))
px.histogram(df, x='diagnosis')
# plt.show()
```

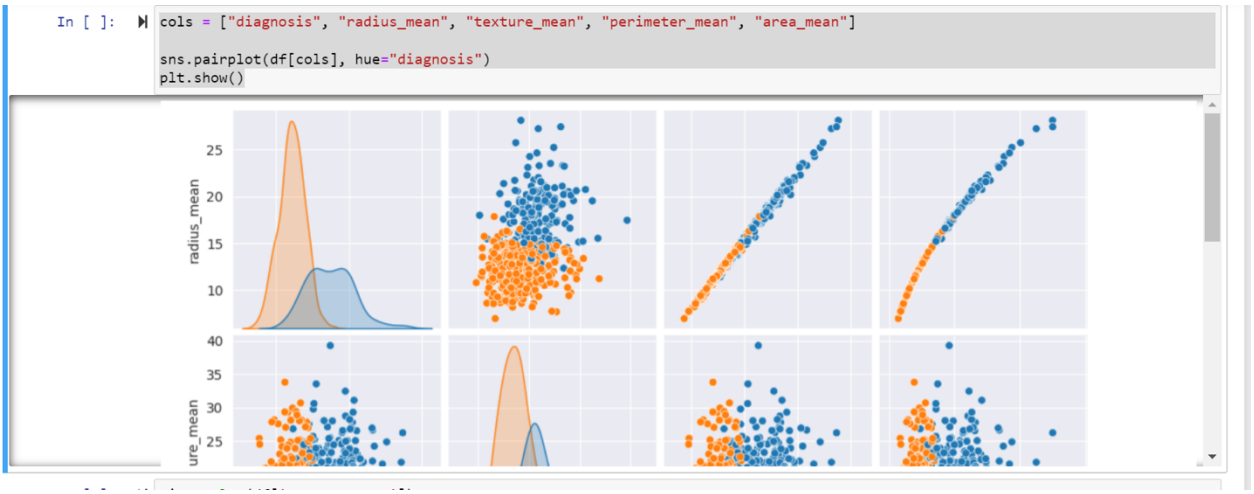


Fig 3.7

To visualize the pairwise relations between each subset in a data frame, we have used the pair plot (Fig. 3.7). The pair plot shows scatterplots of each other, along with the distribution of the plot. This is based on the diagnosis data frame.Following is a visualization of the distribution of texture mean and the radius mean feature in the data set (Fig. 3.8). This is used to identify the patterns in the datasets. It is also used to identify the relations between the features and outcome variable that is cancerous spreading or non-cancerous non-spreading.

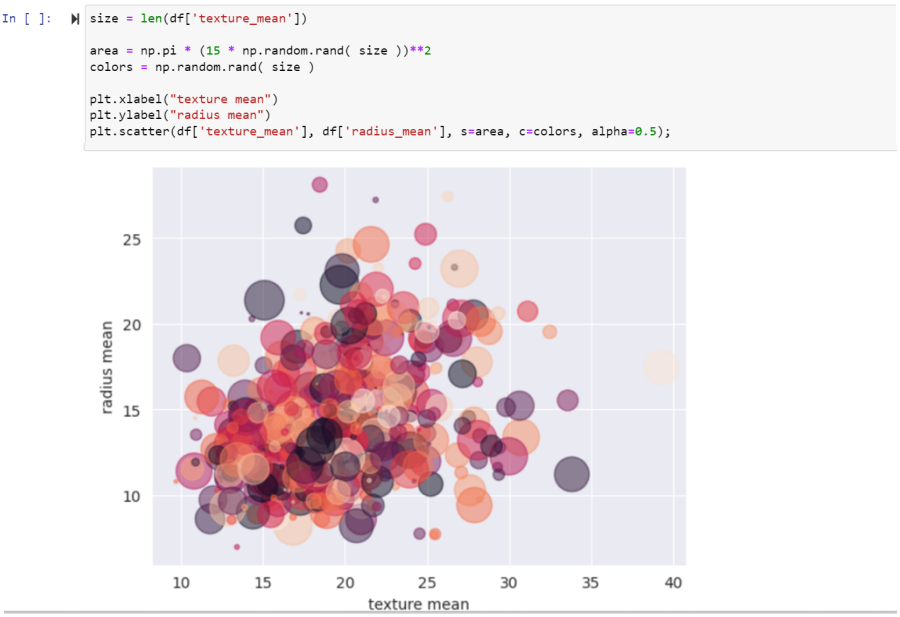
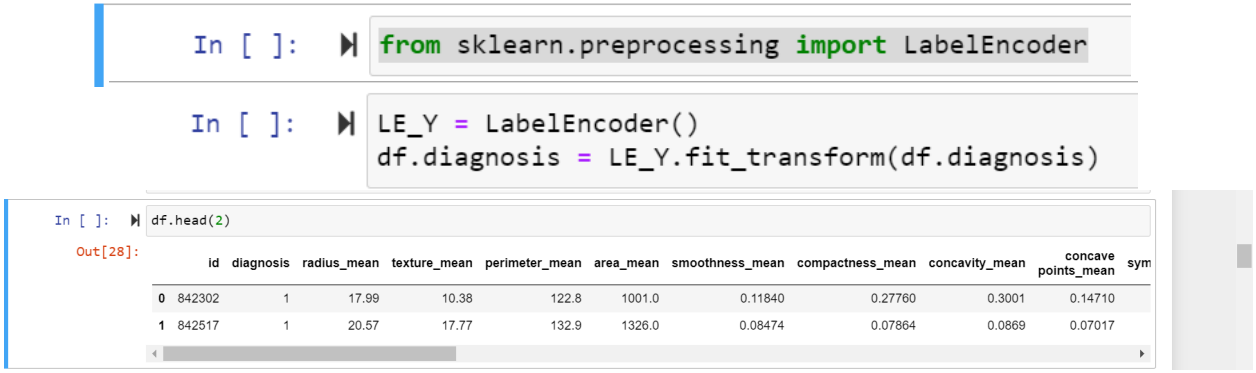


Fig. 3.8

In the above step, we are importing the *label encoder* library (Fig. 3.9). This is one of the most important libraries in *sklearn*. This is used to convert text data into numerical data. As we know that the label encoder is the library that is used to convert the text data into the numerical data

the above code. This is the important part of the code where it is used to convert the breast cancer data set into the numerical forms using the logistic regression method. It is going to predict whether the tumor is non -cancerous non spreading or cancerous and spreading. It is going to give the output as 0 or 1. As discussed previously, df.head() is used to display the first few parts of the columns of the dataset.



	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	sym
0	842302	1	17.99	10.38	122.8	1001.0	0.11840	0.27760	0.3001	0.14710	
1	842517	1	20.57	17.77	132.9	1326.0	0.08474	0.07864	0.0869	0.07017	

Fig. 3.9

Now, let us now understand dataset. The dataset contains different columns such as diagnosis, *radius_mean,texture_mean,perimeter_meanarea_mean,smoothnes_mean,compactness_mean, concavity_mean*, etc. These features can be input for the input regression this is used to predict whether the tumor is non-cancerous non-spreading or cancerous and spreading.

- radius_mean: the mean distances from center to points in the tumor perimeter.
- texture_mean: the mean of the grayscale values
- perimeter_mean: the mean of the perimeter values
- area_mean: the mean of the area of the tumor
- smoothness_mean: local variation
- compactness_mean: square of the tumor perimeter divided by the area minus 1.

These are important columns in the datasets. However, there are different columns in the datasets such as concavity_mean, concave_points mean, etc. This dataset also has different columns which also calculate standard errors of the radius, perimeters areas, etc. Fig. 3.10 contains 2 lines of code. This code is used to get the total number of observations in the dataset. This clarifies the characteristics of the dataset so that can be used in training the machine using the logistic regression algorithm.

We now generate the correlation matrix (Fig. 3.11) from the subset of the dataset. This step is used to find out multicollinearity and avoid them as much as possible when performing the logistic regression algorithm.

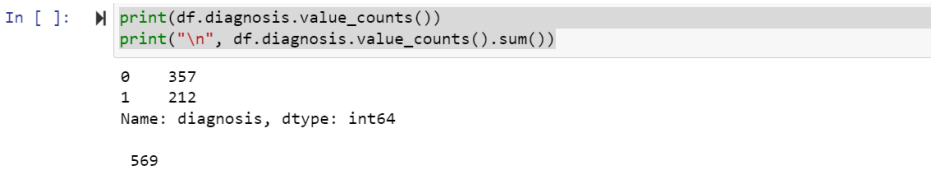


Fig. 3.10

In [ ]:
```
cols = ['diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean']
print(len(cols))
df[cols].corr()
```

```
11
```

Out[30]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	poin
diagnosis	1.000000	0.730029	0.415185	0.742636	0.708984	0.358560	0.596534	0.696360	
radius_mean	0.730029	1.000000	0.323782	0.997855	0.987357	0.170581	0.506124	0.676764	
texture_mean	0.415185	0.323782	1.000000	0.329533	0.321086	-0.023389	0.236702	0.302418	
perimeter_mean	0.742636	0.997855	0.329533	1.000000	0.986507	0.207278	0.556936	0.716136	
area_mean	0.708984	0.987357	0.321086	0.986507	1.000000	0.177028	0.498502	0.685983	
smoothness_mean	0.358560	0.170581	-0.023389	0.207278	0.177028	1.000000	0.659123	0.521984	
compactness_mean	0.596534	0.506124	0.236702	0.556936	0.498502	0.659123	1.000000	0.883121	
concavity_mean	0.696360	0.676764	0.302418	0.716136	0.685983	0.521984	0.883121	1.000000	
concave points_mean	0.776614	0.822529	0.293464	0.850977	0.823269	0.553695	0.831135	0.921391	
symmetry_mean	0.330499	0.147741	0.071401	0.183027	0.151293	0.557775	0.602641	0.500667	
fractal_dimension_mean	-0.012838	-0.311631	-0.076437	-0.261477	-0.283110	0.584792	0.565369	0.336783	

Fig 3.11

In [ ]:
```
plt.figure(figsize=(12, 9))

plt.title("Correlation Graph")

cmap = sns.diverging_palette( 1000, 120, as_cmap=True)
sns.heatmap(df[cols].corr(), annot=True, fmt='.1%',  linewidths=.05, cmap=cmap);
```

We then produce the heatmap of the correlation matrix that is generated in the above block. This is used to identify the relationships between the features in the dataset. (Fig. 3.12)

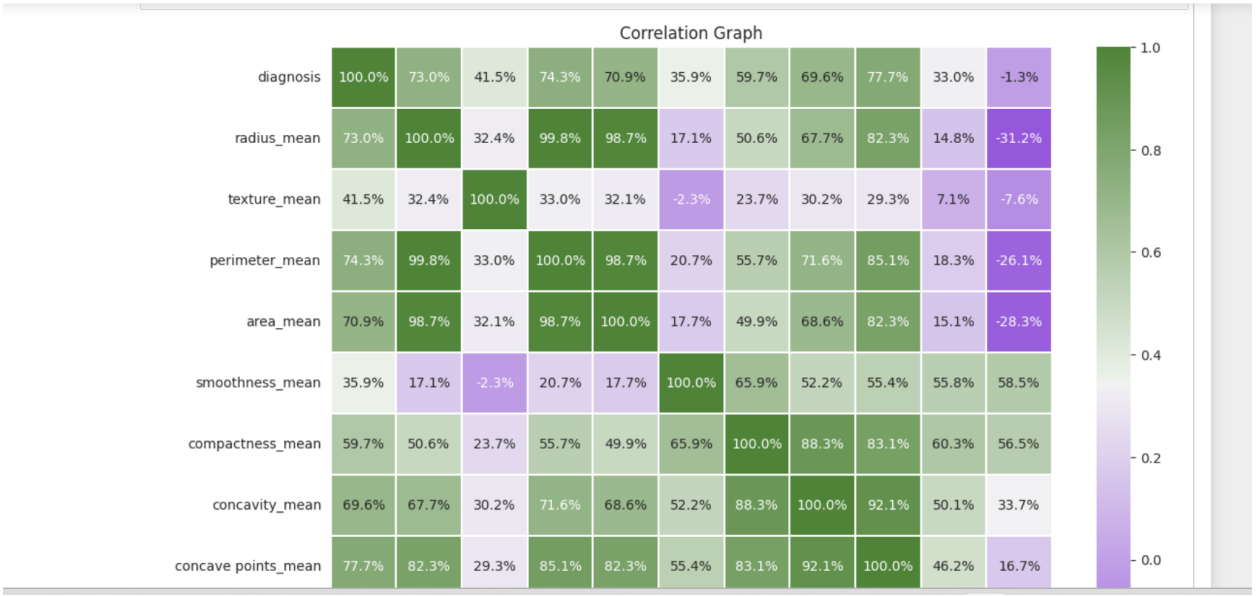


Fig. 3.12

3.2.3 Training Models

This is the first step in importing the machine learning models. This requires in importing many algorithms such as *LogisticRegression*, *DecisionTreeClassifier*, *RandomForestClassifier*, *GaussianNB* (Fig. 3.13), and some other important libraries that are used to perform machine learning logistic regression algorithm.

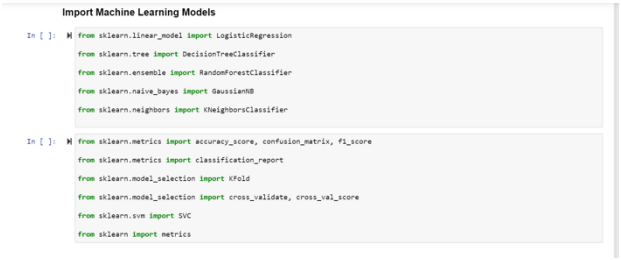


Fig. 3.13

The prediction feature is the to predict the targeted variables called *targeted_features*. *len* is used to determine the length of the list that is 6 here (Fig. 3.14).

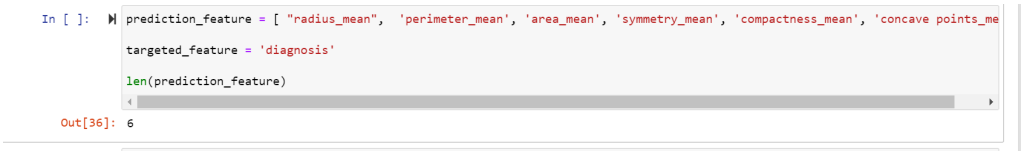


Fig. 3.14

The output of the code block will be the values of the "diagnosis" column from the DataFrame df, displayed as both a Pandas Series object and a NumPy array (Fig. 3.15).

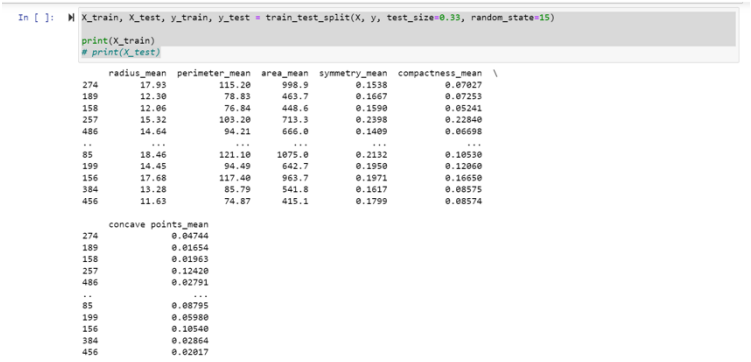


Fig. 3.15

X_train and X_test are the feature data for the training and testing sets, respectively. The feature data consists of the independent variables or predictors, in this case, the values of certain attributes of breast cancer cells such as radius, perimeter, area, symmetry, etc. y_train and y_test are the target data for the training and testing sets, respectively. The target data consists of the dependent variable or the variable we want to predict, in this case, the diagnosis of breast cancer cells, which can be either malignant or benign.

```
In [ ]:  # Scale the df to keep all the values in the same magnitude of 0 -1

         sc = StandardScaler()

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

```
In [ ]:  def model_building(model, X_train, X_test, y_train, y_test):
             """

             Model Fitting, Prediction And Other stuff
             return ('score', 'accuracy_score', 'predictions' )
             """

             model.fit(X_train, y_train)
             score = model.score(X_train, y_train)
             predictions = model.predict(X_test)
             accuracy = accuracy_score(predictions, y_test)

             return (score, accuracy, predictions)
```

```
In [ ]:  models_list = {
             "LogisticRegression" :  LogisticRegression(),
             "RandomForestClassifier" :  RandomForestClassifier(n_estimators=10, criterion='entropy', random_state=5),
             "DecisionTreeClassifier" :  DecisionTreeClassifier(criterion='entropy', random_state=0),
             "SVC" :  SVC(),
         }

         # print(models_list)
```

Fig. 3.16

The code in Fig. 3.16 defines a function called *model_building* which fits a given model on the training data, calculates the training score, predicts the target variable for the testing data, calculates the accuracy score, and returns the score, accuracy, and predictions as output. The code also defines a dictionary called *models_list* which contains four different machine learning models: *LogisticRegression*, *RandomForestClassifier*, *DecisionTreeClassifier*, and SVC. It also prints out the list of model objects in the dictionary. The first two print statements convert the dictionary keys and values into lists and print them. The commented line uses the zip function to create a list of tuples where each tuple contains a model name and its corresponding model object from the *models_list* dictionary.The cm_metrix_graph function takes a confusion matrix (cm) as input, and it uses the seaborn library to create a heatmap of the confusion matrix. The annot=True parameter displays the values of the matrix inside the heatmap. The fmt="d" parameter formats the values as integers. Finally, the function shows the plot using the plt.show() function.

```
In [ ]:  df_prediction = []
         confusion_matrixs = []
         df_prediction_cols = [ 'model_name', 'score', 'accuracy_score' , "accuracy_percentage"]

         for name, model in zip(list(models_list.keys()), list(models_list.values())):

             (score, accuracy, predictions) = model_building(model, X_train, X_test, y_train, y_test )

             print("\n\nClassification Report of '"+ str(name), "'\n")

             print(classification_report(y_test, predictions))

             df_prediction.append([name, score, accuracy, "{0:.2%}".format(accuracy)])

             # For Showing Metrics
             confusion_matrixs.append(confusion_matrix(y_test, predictions))


         df_pred = pd.DataFrame(df_prediction, columns=df_prediction_cols)
```

Refer to Fig. 3.17 for this part of the implementation. We iterate over the models in the *models_list* dictionary and calls the *model_building* function for each model. The *model_building* function fits the model on the training data, predicts the target variable for the testing data, calculates the accuracy score and the training score of the model, and returns them.For each model, the code prints the classification report and adds the model's name, training score, accuracy score, and accuracy percentage to the df_prediction list.The code also creates a list called confusion_matrixs which contains the confusion matrix for each model.Finally, the code creates a pandas DataFrame called df_pred with the columns ['model_name', 'score', 'accuracy_score', "accuracy_percentage"] and populates it with the df_prediction list.

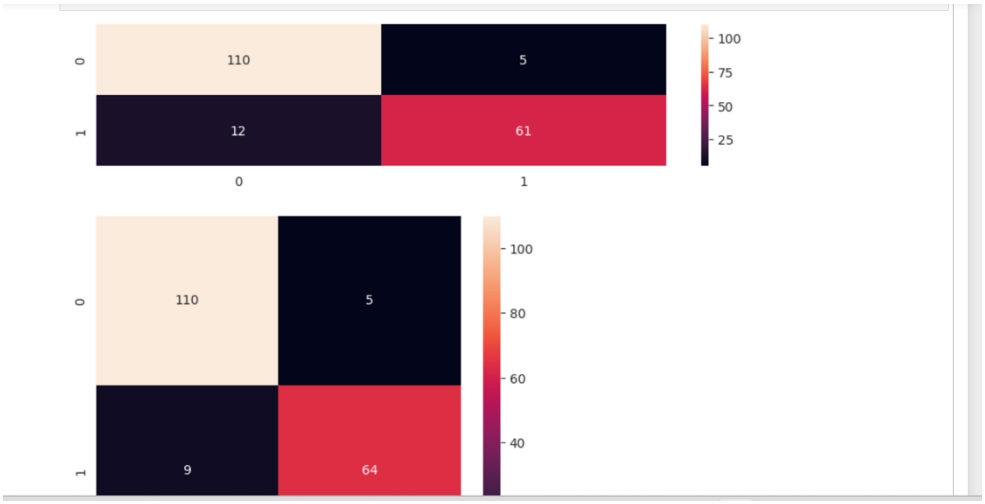
```
Classification Report of 'LogisticRegression '

              precision    recall  f1-score   support

           0       0.90      0.96      0.93       115
           1       0.92      0.84      0.88        73

    accuracy                           0.91       188
   macro avg       0.91      0.90      0.90       188
weighted avg       0.91      0.91      0.91       188


Classification Report of 'RandomForestClassifier '

              precision    recall  f1-score   support

           0       0.92      0.96      0.94       115
           1       0.93      0.88      0.90        73

    accuracy                           0.93       188
   macro avg       0.93      0.92      0.92       188
weighted avg       0.93      0.93      0.93       188


Classification Report of 'DecisionTreeClassifier '

              precision    recall  f1-score   support

           0       0.90      0.96      0.93       115
```

Fig. 3.17

Here, each pass in the loop generates one heatmap of the each that represents the confusion matrix. The heatmaps are generated side by side (Fig. 3.18, 3.19). The output of the code is attached below.

```
In [ ]:
plt.figure(figsize=(10, 2))
# plt.title("Confusion Metric Graph")


for index, cm in enumerate(confusion_matrixs):

#     plt.xlabel("Negative Positive")
#     plt.ylabel("True Positive")



    # Show The Metrics Graph
    cm_metrix_graph(cm) # Call the Confusion Metrics Graph
    plt.tight_layout(pad=True)
```

Fig. 3.18



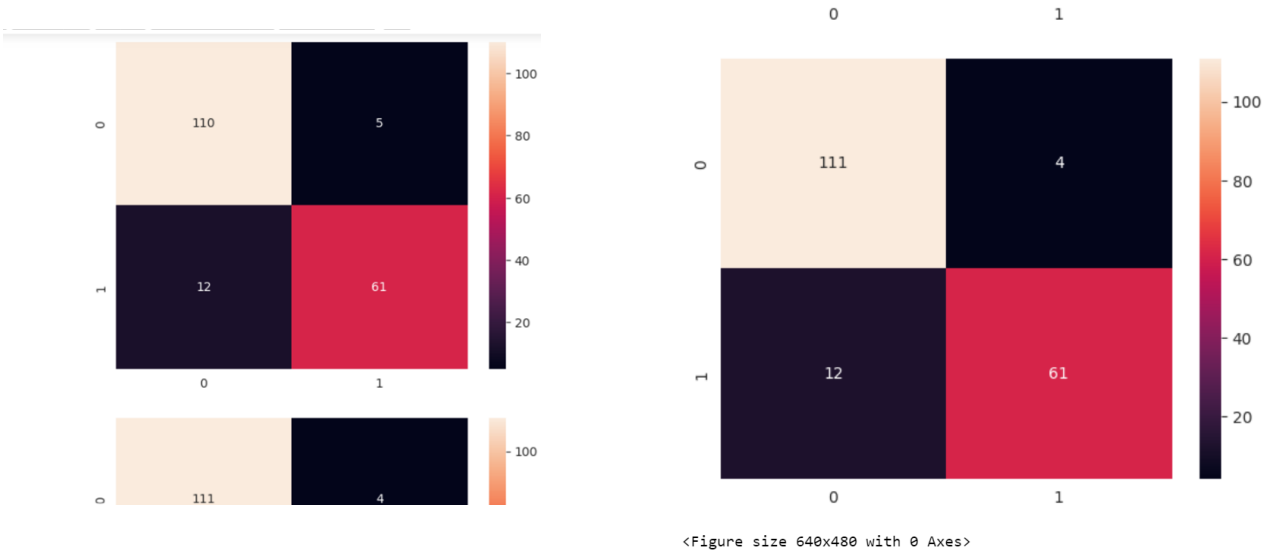


Fig 3.19

Finally, Fig. 3.20 depicts the last step in the breast cancer prediction. We have a parameter *df_pred* to store the prediction results and evaluation metrics.

```
In [ ]: df_pred
```

Out[48]:

	model_name	score	accuracy_score	accuracy_percentage
0	LogisticRegression	0.916010	0.909574	90.96%
1	RandomForestClassifier	0.992126	0.925532	92.55%
2	DecisionTreeClassifier	1.000000	0.909574	90.96%
3	SVC	0.923885	0.914894	91.49%

Fig. 3.20

3.3 Menstrual Cycle Calendar

3.3.1 Preliminary results

A simple menstrual cycle calendar has been implemented using python [12]. Following are a few snippets of the user interface that has been implemented so far.

The first page that a user sees is the Login page. (Fig. 3.21). Here we can see the login page where in the left side of the page we can see the login option, after login, cycle prediction and model prediction options. When we click on the login main option, we can enter the username and password credentials to login into the application. When we click on the after-login page, we can see the application where we have breast cancer prediction and menstrual cycle prediction button to predict and the logout option to logout of the page. A header message has been displayed “welcome to sakhi” once we login.

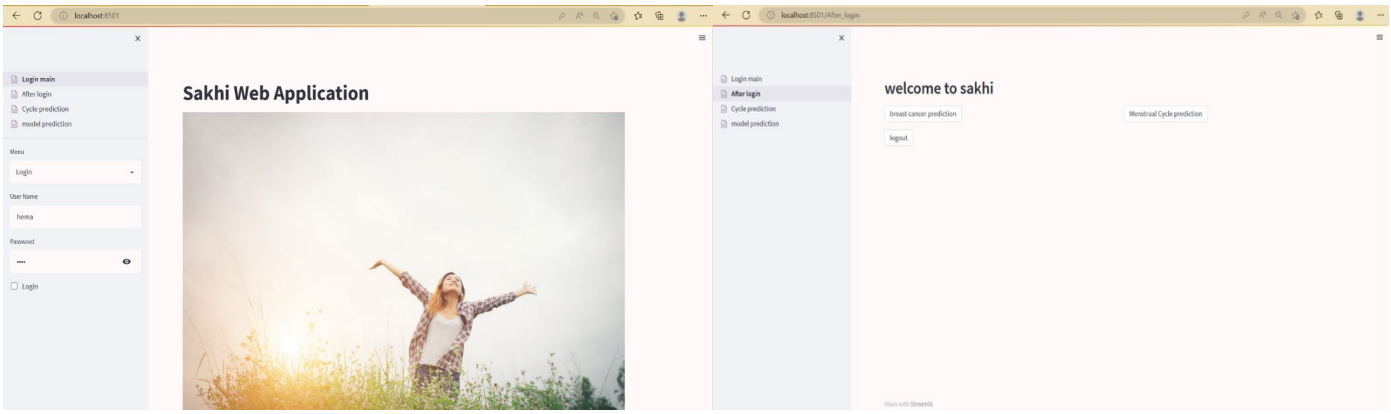


Fig. 3.21 (Before and After logging in)

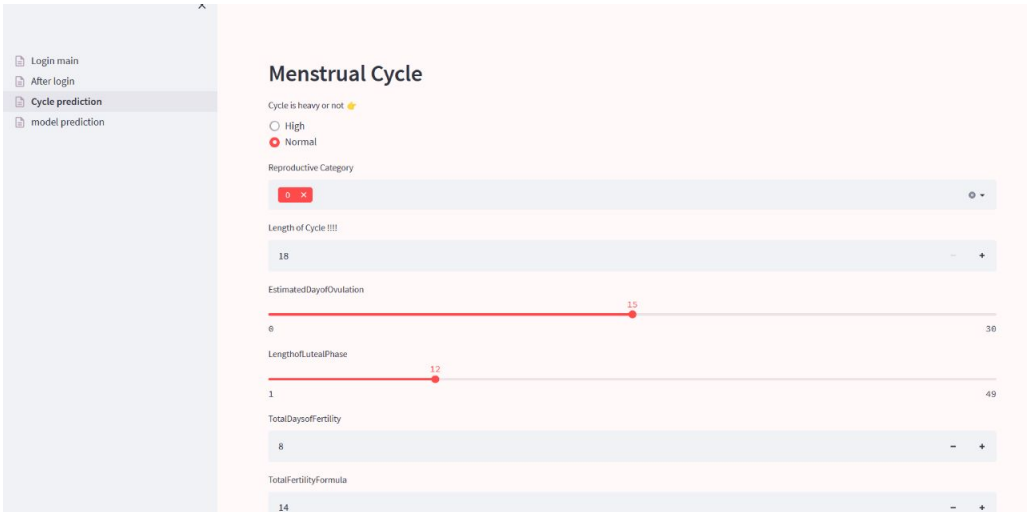


Fig: 3.22

When we select the cycle prediction option, we can see the options where we must enter the details to predict the data (Fig. 3.22). Firstly, we can select whether the flow is normal or high. We have an option to enter the data for the reproductive category. The length of the cycle defines the no of days the next cycle would occur. The estimated day of ovulation based on the length of the menses the ovulation occurs in general so we can enter the data for the date by dragging the slider to and from. Length of Luteal phase the range starts from 1 to 49 as per the dataset. Total days of fertility can be mentioned we can enter the value by clicking on the '- ' and '+' signs.

Total Fertility Formula can be mentioned in the data by clicking on '-' and '+'. We can enter the start date and the end date of the period by selecting the month, year, and date. We can see the '-' and '+' over the date if we click on the add button we can add the period date of the previous months data as well.

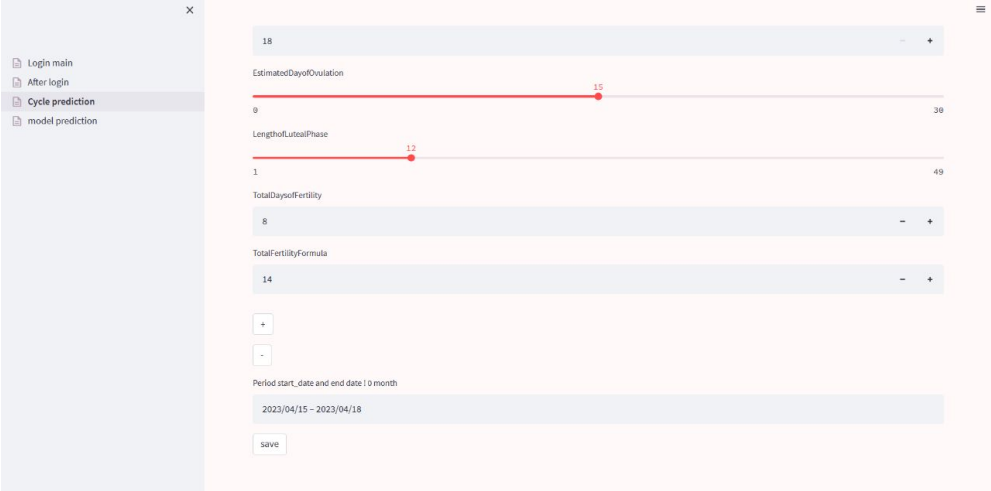


Fig: 3.23

In Fig. 3.24, we can see the calendar option to enter the start and the end date of the period. Based on the prediction model the period is calculated. We can the display number (period date) over the application, and we can also see the message how many days the next menstrual cycle would occur.

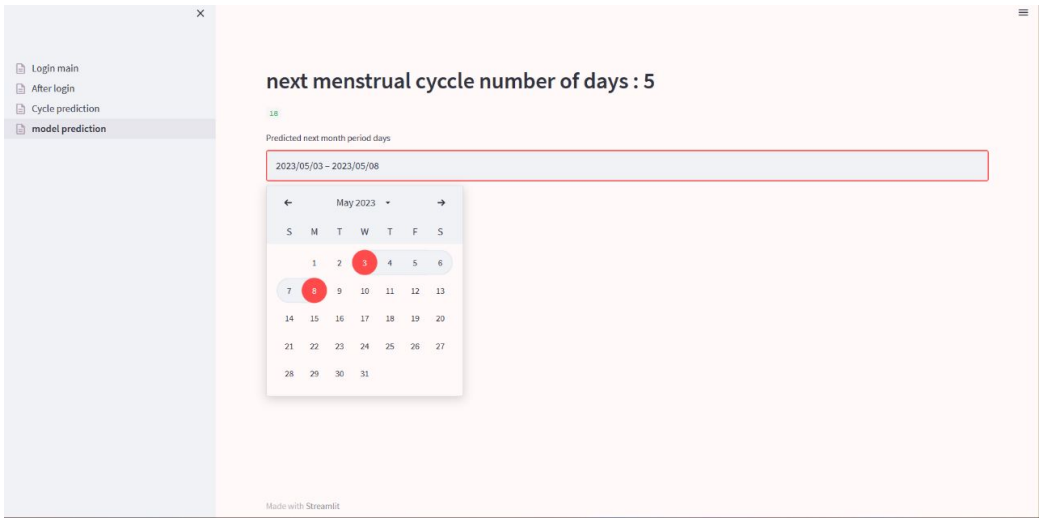


Fig: 3.24

3.3.2 Technical Implementation details

Login_page.py: We have used *streamlit* framework to use the coding part. Def *usertable_creation ()* is used to create table if the table is not existing. It verifies the username and the password in the database. *Def verify_user()* the function has the parameters user and *pwd*. The *e.execute* will verify the user in the database and if it is present then the data is being fetched by using *c.fetchall().*Def *view_all_users()* is used to retrieve the data users present in the database. *Def switch_page()* this function is used to switch the multiple pages from one to another. *Def*

standardize_name is used to return the string name *lower.replace()* is used to replace the value with "_". Refer Fig. 3.25 for the code.

```
import streamlit as st
import sqlite3
import streamlit_login
import streamlit as st;
from httpx_oauth.clients.google import GoogleOAuth2
import asyncio
import webbrowser
from numpy import void
from pages import After_login;
from streamlit import _RerunData;
from streamlit import _RerunException;
import streamlit as st
from PIL import Image
conn = sqlite3.connect('User.db')
c = conn.cursor()
#st.set_page_config(page_title='Multipage app',layout='wide');


def UserTable_creation():
        c.execute('CREATE TABLE IF NOT EXISTS users(username TEXT,password TEXT)')

def Verify_user(user,pwd):
        c.execute('SELECT * FROM users WHERE username =? AND password = ?',(user,pwd))
        data = c.fetchall()
        return data

def view_all_users():
        c.execute('SELECT * FROM users')
        data = c.fetchall()
        return data
def switch_page(page_name: str):

    from streamlit.source_util import get_pages
    def standardize_name(name: str) -> str:
        return name.lower().replace("_", " ")


    page_name = standardize_name(page_name)

    pages = get_pages("streamlit_app.py")  # OR whatever your main page is called
```

Fig. 3.25

*Def main() (*Fig. 3.26) is the function where we have declared the menu and the login details. *St.title()* is used to display the message written in the argument. *Image.open()* will open the image present declared here. *St.image()* is present where we have written the caption as "women wellness". *Main,signup,login* and google options are declared in the Menu. We can select any option from the menu by using *st.sidebar.selectbox()* .

```
    pages = get_pages("streamlit_app.py")  # OR whatever your main page is called

    for page_hash, config in pages.items():
        if standardize_name(config["page_name"]) == page_name:
            raise _RerunException(
                _RerunData(
                    page_script_hash=page_hash,
                    page_name=page_name,
                )
            )

    page_names = [standardize_name(config["page_name"]) for config in pages.values()]

    raise ValueError(f"Could not find page {page_name}. Must be one of {page_names}")


def main():
        st.title(" Sakhi Web Application");
        image = Image.open('health-wellbeing-womens-health-1024x683.jpg')
        st.image(image, caption='Women Wellness')
        menu = ["Main","SignUp","Login","Google"]
        choice = st.sidebar.selectbox("Menu",menu)
        if choice == "Home":
                st.subheader("Login into Ur Account")
        elif choice == "Login":
```

Fig. 3.26

If-else statements has been used here if we select home option, we can see the text header as “Login into your account” elif we can choose Login we can login with username and password.

```
                        UserTable_creation()
                        if Verify_user(uname,pwd):
                                st.success("Login Successfully"+uname);
                                #OPEN OR GO TO THE NEW PAGE WHERE U WILL HAVE NEW PAGES HERE AND THEN WE HAVE TO GO TO THAT PAGE.
                                # with open("image.png", "rb") as file:
                                #       btn = st.download_button(label="Download image",data=file,file_name="flower.png",mime="image/png");
                                # st.success("Image downloaded successfully!!!!!!!!!!!!!");
                                #create a sessio variable we
                                st.session_state["login_state"]="SUCCESS";
                                #After_login.login_system(uname);
                                switch_page('After_login');
                        else:
                                st.warning("Incorrect Username/Password");
                                st.session_state["login_state"] = "FAILED";
        elif choice=="Google":
                        st.title("Streamlit Google Login");
                        st.write(streamlit_login.get_login_str(), unsafe_allow_html=True)
                        st.info("Logged in successfully!!!!!!!!!");

                        with open("pages/image.png", "rb") as file:
                                btn = st.download_button(label="Download image", data=file, file_name="flower.png", mime="image/png");
                        #st.success("Image downloaded successfully!!!!!!!!!!!!!");
                        switch_page('After_login');
                        st.balloons();
        elif choice == "SignUp":
                st.subheader("New User Creation")
                nuser = st.text_input("Username");
                npassword = st.text_input("Password",type='password');
                if st.button("Signup"):
                        UserTable_creation();
                        c.execute('INSERT INTO users(username,password) VALUES (?,?)',(nuser,npassword));
                        st.balloons();
                        st.info("User Created!!!!!")
                        conn.commit();

if __name__ == '__main__':
        #reset the cache...
        st.session_state["login_state"] = "FAILED";
        st.set_page_config(page_title='Multipage app', layout='wide');

        main()
```

Fig. 3.27

Verify_user() will verify the login details present in the database. *St.success()* will retrieve the message as “Login successfully” if it is logged in successfully. *Switch_page()* we have another file name to switch the page once it is logged in else we will get a error message as “Incorrect username/password”. We can also login by using Google authentication. St.title() will display the title header message. After login we can see the message as login successfully and we can also see signup option in the menu *st.subheader()* we will get header message as new user creation. User input will be given as username and password and *st.button()* is used to display the button signup. *C.execute()* will insert the details into the database and enter the details. *Conn.commit ()* will use to commit the code and will save the details once it is submitted, we can see the balloons over the screen which means the login is successful.

After_login.py: *Toml* is being imported to configure the files in a format. *_rerundata* is used to rerun the files while switch from page to page to trigger the data based on external data. _rerun exception is used to hand the unnecessary exceptions raised during the execution try catch blocks are used to handle the exceptions. *Def switch_pages()* is used to switch the pages from one to another. *Page_hash* will help to interact with the indexes and widgets within in the code. When we give the file name the *switch_page ()* will redirect to the next page whatever we have given (Fig. 3.28).

(Fig. 3.29) Def *login_system()* is used where the login details are present. *St.header()* is used to display the header message as “welcome to sakhi”. *St.button()* is used to display the button where the breast cancer prediction context is being displayed. When we click on the button the *switch_page()* will redirect to the *cycle_prediction* page. The other button is declared as menstrual cycle prediction using *st.button().*

```
import streamlit as st;
import toml;
from streamlit import _RerunData
from streamlit import _RerunException
#from streamlit.script_runner import RerunException
from streamlit.source_util import get_pages

def switch_page(page_name: str):
    def standardize_name(name: str) -> str:
        return name.lower().replace("_", " ")


    page_name = standardize_name(page_name)

    pages = get_pages("streamlit_app.py")  # OR whatever your main page is called

    for page_hash, config in pages.items():
        if standardize_name(config["page_name"]) == page_name:
            raise _RerunException(
                _RerunData(
                    page_script_hash=page_hash,
                    page_name=page_name,
                )
            )

    page_names = [standardize_name(config["page_name"]) for config in pages.values()]
```

Fig. 3.28

```
    raise ValueError(f"Could not find page {page_name}. Must
def login_system(uname):
    st.header("welcome to sakhi ");

    col1 , col2 = st.columns(2);
    with col1:
        if(st.button("breast cancer prediction ")):
            switch_page('Cycle_prediction')


    with col2:
        if(st.button("Menstrual Cycle prediction")):
            switch_page('Cycle_prediction')

    if(st.button("logout")):
        st.write("logged out successfully!");
        switch_page("Login_main");
if __name__ == '__main__':
        login_system("Hema");
```

Fig. 3.29

Cycle_prediction.py: Based on specific characteristics, the provided Python code creates a *Streamlit* web application that forecasts a woman's menstrual cycle. Cycle (Heavy or Normal), Reproductive Category, Cycle Length, Estimated Day of Ovulation, Luteal Phase Length, Total Days of Fertility, and Total Fertility Formula are among the characteristics. Additionally, users of the application can enter information from the previous three months to determine the typical menstrual cycle length. In the first line of the code, the relevant libraries are imported, including *Streamlit* for building the web application, NumPy for performing mathematical operations, datetime for working with date and time data, and *StandardScaler* and pickle from the scikit-learn package. The application's header is created using Streamlit's header method and has the heading "Menstrual Cycle."

```
import streamlit as st;
import numpy as np;
import datetime;
from sklearn.preprocessing import StandardScaler
st.header("Menstrual Cycle ");
import pickle;
from streamlit import _RerunData
from streamlit import _RerunException
#from streamlit.script_runner import RerunException
from streamlit.source_util import get_pages


def switch_page(page_name: str):
    def standardize_name(name: str) -> str:
        return name.lower().replace("_", " ")


    page_name = standardize_name(page_name)

    pages = get_pages("streamlit_app.py")  # OR whatever your main page is called

    for page_hash, config in pages.items():
        if standardize_name(config["page_name"]) == page_name:
            raise _RerunException(
                _RerunData(
                    page_script_hash=page_hash,
                    page_name=page_name,
                )
            )

    page_names = [standardize_name(config["page_name"]) for config in pages.values()]

    raise ValueError(f"Could not find page {page_name}. Must be one of {page_names}")
```

Fig. 3.30

(Fig. 3.30) *"switch_page"* is the next function specified in the code. The name of the page to switch to is the only input for this function. Within the function, a nested function called *"standardize_name"* is constructed, which accepts a string as input and outputs the same string in lower case by replacing underscores with spaces. The application's whole page collection can be retrieved using the *'get_pages'* function from the *Streamlit source_util* package. To determine whether pages have names that match the input parameter, the retrieved pages are iterated over. The application reruns with the provided page if there is a match, and a *Streamlit*-specific exception called "_RerunException" is raised. A *"ValueError"* is raised with a message stating that the input parameter is invalid if there is no match.

Predict_model is the next function, and it only accepts one parameter, called "*X_test.*" This function standardizes the input data using "*StandardScaler*" from the scikit-learn library before loading a saved machine learning model from a file using the "pickle" library. After applying the "predict" function to the pre-processed input, the outcome is saved in the variable "result." (Fig. 3.31) Following that, the anticipated outcome is shown on the *Streamlit* app using the "header" function and is also saved in the "result" value of the Streamlit session_state dictionary. Finally, *"model_prediction,"* the name of the page that will be shown next, is passed as an argument to the *"switch_page"* method. The remaining code is made up of other *Streamlit* input components, including *"radio," "multiselect," "number_input,"* and "*select_slider,"* that let users submit data for the different aspects of the menstrual cycle. The input data is added to a list called *"X_new,"* which is then used as input to the *"predict_model"* function.

```
def predict_model(X_test):
    filename = 'finalized_model.sav'
    loaded_model = pickle.load(open(filename, 'rb'))
    print("x_test : ",X_test);
    scaler = StandardScaler()
    X_test = scaler.fit_transform(X_test);
    result = loaded_model.predict(X_test)
    print(result[0]);
    print(" prediction model ")
    st.header(result[0]);
    st.session_state['result']=result[0];
    print(st.session_state);
    switch_page("model_prediction")

#input the data for prediction;
#st.text_input("Cycle is heavy or not");
X_new = [];
cycle = st.radio("Cycle is heavy or not 👉",options=["High", "Normal"]);

if cycle == "High":
    X_new.append(1);
else:
    print(" cycle is normal !!!");
    X_new.append(0);

#ReproductiveCategory

Reprod_category = st.multiselect('Reproductive Category ',[0,1,2,9]);
print(Reprod_category);
if len(Reprod_category)>=1:
    X_new.append(Reprod_category[0]);
```

Fig. 3.31

The code additionally enables users to enter data for the previous three months using the *'date_input'* method in addition to the functionality stated above. Using the *'st.columns'* method, a variable number of text boxes is generated, allowing users to add or delete text boxes as necessary. The input data for the three months is saved in a list called *"period_dates"* when the *"save"* button is clicked. The average of the lengths is added to the *'X_new'* list after each menstrual cycle is measured and the length is determined by deducting the end date from the start date.

```
#LengthofLutealPhase
Length_of_Luteal_Phase = st.select_slider('LengthofLutealPhase',options=[i for i in range(1,50)]);
X_new.append(Length_of_Luteal_Phase);


#TotalDaysofFertility
TotalDaysofFertility = st.number_input('TotalDaysofFertility',min_value=0,max_value = 27, step=1);
X_new.append(TotalDaysofFertility);


#TotalFertilityFormula
TotalFertilityFormula = st.number_input('TotalFertilityFormula',min_value=3,max_value = 40, step=1);
X_new.append(TotalFertilityFormula);


#LengthofMenses = st.select_slider('LengthofMenses',options=[i for i in range(2,15)]);
col1 , col2 = st.columns(2);
if 'n_rows' not in st.session_state:
    st.session_state.n_rows = 1
add = st.button(label="+")
sub = st.button(label="-")
with col1:
    if add:
        st.session_state.n_rows += 1
        st.experimental_rerun()
with col2:
    if sub:
        if st.session_state.n_rows>0:
            st.session_state.n_rows -= 1
            st.experimental_rerun()
        else:
            st.warning("all the text boxes were deleted")
period_dates=[];
for i in range(st.session_state.n_rows):
    start_date = st.date_input('Period start_date and end date ! '+str(i)+" month ",value=[datetime.date(2023, 4, 15), datetime.date(2023, 4, 20)]);
    period_dates.append(start_date);
```

```
st.session_state['dates_months'] = period_dates;
#get the average of those mensus cycle
if st.button('save'):
    each_month =[];
    #save the button
    for i in period_dates:
        lm = i[1] - i[0];
        LengthofMenses = lm.days;
        each_month.append(lm.days);
    print(X_new);
    X_new.append(sum(each_month)/len(each_month));
    print(" inside the period dates !!!")
    print(X_new);
    X_test = [X_new];
    X_test = np.array(X_test);
    print(X_test);
    predict_model(X_test);
```

Fig. 3.32

Model_prediction.py: Pandas' library is used to handle the datasets in the data structures. Date time is being imported to retrieve the date and time. *_rerun* and the *_rerun* exceptions are used to handle the exceptions raised. *St.session_state()* is used to store the variables between the pages. The *dates_months* is calculated by considering the *length_of_cycles* and the start date is being printed. *Datetime.timedelta()* is used to add the mathematical function when it is time format. The resulted end date would be printed. *St.date_input()* is used to give the date manually for predicting the month starting and ending date.

```
import pandas as pd;
import streamlit as st;
import datetime
from streamlit import _RerunData
from streamlit import _RerunException
#from streamlit.script_runner import RerunException
from streamlit.source_util import get_pages

st.header(" next menstrual cycle number of days : "+str(st.session_state['result']));
#st.session_state['dates_months'] = period_dates;
#date_input
st.session_state['length_of_cycle']
start_date = st.session_state['dates_months'];
print(start_date)
lengthofcycle=st.session_state['length_of_cycle'];
next_month_start_date = start_date[0][0]+datetime.timedelta(days=lengthofcycle);
next_month_end_date = next_month_start_date+datetime.timedelta(days =int(st.session_state['result']));

sd = st.date_input('Predicted next month period days ',value=[next_month_start_date,next_month_end_date]);
```

Fig. 3.33

3.4 Menstrual cycle prediction with binary classification using Logistic regression

The Python script demonstrates the Menstrual cycle prediction with binary classification using logistic regression on a dataset.We first need to import certain libraries to make code function:

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

Fig. 3.34

- *Pandas as pd:* It is used for data manipulation and analysis. The above code reads the CSV file as a data frame and performs various operations [11].
- *Numpy as np:* It is used for numerical computing. In the code, it converts empty strings to NaN values.

```
In [ ]:
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
```

Fig. 3.35

- *train_test_split from sklearn.model_selection:* It is used to split the data into training and testing sets for model training and evaluation, respectively.
- S*tandardScaler from sklearn.preprocessing:* It is used for feature scaling, regulates the distribution of data across features so they have the same mean and standard deviation.
- *LogisticRegression from sklearn.linear_model:* This class is used for training a logistic regression model.

In [ ]:
```
# load the dataset
data = pd.read_csv("/content/data.csv")
# preprocessing
if 'CycleNumber' in data.columns:
    data.drop(['ClientID', 'CycleNumber'], axis=1, inplace=True)
else:
    data.drop('ClientID', axis=1, inplace=True)
# Replace empty strings with NaN values
data = data.replace('', np.nan)
```

Fig. 3.36

Fig 3.36 shows loading a CVS file containing data and performs some preprocessing steps on it. Using the *read_csv()* method the data is loaded into Pandas DataFrame. The code then checks for the column in the dataset with the name *cycleNumber* if it exits then columns ClientID are dropped from the dataframe using *drop ()*.If *cycleNumber* itself does not exist only ClientID column will be dropped.Code can replace empty values with *NaN* values using the *replace ()* method which is in the *Numpy* Library. This step helps in getting rid of the empty strings which causes issues with data analysis and modelling.

In [ ]:
```
# Convert all columns to numeric data type
data = data.apply(pd.to_numeric)

# Get the features and target variable
X = data.iloc[:, :-1].values
y = data.iloc[:, -1].values

# Scale the features
scaler = StandardScaler()
X = scaler.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# fit the model
clf = LogisticRegression()
clf.fit(X_train, y_train)
```

Fig. 3.37

iloc method is used to select columns by index position. In Fig. 3.37, all columns except the last one are assigned to feature matrix X and the last column is assigned to the target vector Y. The *StandardScaler()* is from sci-kit-learn library this plays a vital role in machine learning algorithms as it regulates the distribution of the data across features so they have the same mean and standard deviation.Then *train_test_split()* from scikit learn library splits the dataset into training and testing sets with the *parameter test_size* it governs the proportion of the dataset to include

in the test split.Random_state is used to maintain the randomly generated values are consistent.This code trains the *LogisticRegression()* from scikit learn on the training data where fit() method is called clf object with the training feature matrix x_train and target vector y_train.

Pickle: It is used for for serialization and deserialization of the python objects here it is used to save and load the train model.

```
In [ ]:  import pickle
```

Fig. 3.38

```
In [ ]:  filename = 'finalized_model.sav'
         pickle.dump(clf, open(filename, 'wb'))

In [ ]:  data.head();
```

Fig. 3.38

In Fig. 3.38, the trained model is saved into a file using pickle module named finalized_model.sav.The dump() function from the pickle is used in clf object which is used to train logistic regression model.Open(flienam,'wb') is used to create the file named finalized_model.sav in current working directory.The data.head() to display the data from dataframe in the first few rows.The following figures are the output of the above code.

Out[15]:

	Group	CycleWithPeakorNot	ReproductiveCategory	LengthofCycle	EstimatedDayofOvulation	LengthofLutealPhase	TotalDaysofFertility	TotalFertilityFormu
0	0	1	0	29	17	12	9	
1	0	1	0	27	15	12	6	
2	0	1	0	29	15	14	5	
3	0	1	0	27	15	12	6	
4	0	1	0	28	16	12	8	

Out[15]:

ithPeakorNot	ReproductiveCategory	LengthofCycle	EstimatedDayofOvulation	LengthofLutealPhase	TotalDaysofFertility	TotalFertilityFormula	LengthofMenses
1	0	29	17	12	9	15	5
1	0	27	15	12	6	13	5
1	0	29	15	14	5	13	5
1	0	27	15	12	6	13	5
1	0	28	16	12	8	14	5

Fig. 3.38

The data in Fig. 3.38 contains Group, Cycle with Peak or Not, Reproductive Category and length of cycle, Estimated Days of ovulation, length of the luteal phase, Total days of fertility, and total fertility Formula and length of Menses.

```
In [ ]:  y_pred = clf.predict(X_test);

In [ ]:  print(y_pred)
```

Fig 3.39

Refer Fig. 3.39. The y_pred is the predicted target variable using the trained logistic regression model from clf on the dataset X_test. Print(y-pred) is used to display the predicted values (Fig. 3.40)

```
[ 5  6  5  4  6  5  4  5  4  5  6  6  5  5  5  5  6  5  5  4  5  5  5  6
  6  5  5  5  5  6  5  5  5  5  5  6  5  5  5  5  6  4  5  7  5  5  5  4
  5  5  6  6  5  5  5  6  6  5  5  6  5  6  5  6  5  5  5  5  5  4  5  5
  5  9  5  5  5  5  5  5  6  5  5  6  6  5  5  6  5  5  4  5  5  5  5  6
  6  5  6  6  5  6  6  5  5  5  5  6  5  6  5  6  5  5  6  5  5  5  5  6
  6  5  5  5  5  5  5  6  5  5  5  6  5  5  5  5  5  5  6  6  5  5  5  5
  5  5  5  5  6  6  5  6  5  5  5  5  6  5  6  6  5  5  5  6  5  6  5  5
  5  5  5  6  5  6  5  4  6  5  6  6  5  5  5  5  5  5  5  5  5  5  5  5
  6  5  5  6  5  5  5  6  5  5  5  5 15  6  5  6  5  5  5  5  4  5  4  6
  5  6  5  5  5  5 15  5  6  5  7  6  5  5  5  5  5  6  6  5  5  4  5  5
  5  5  5  5  5  6  5  5  5  5  5  5  5  6  6  5  5  5  6  5  5  5  5  6
  5  5  6  4  5  5  6  5  5  5  5  5  5  5  5  5  9  5  5  5  5  5  6  5
  6  5  6  5  5  6  5  6  5  5  5  5  5  5  5  5  6  6  6  5  5  6  5  5
  6  5  5  5  6  5  5  6  5  5  4  5  6  5  5  5  5  6  5  6  5  4  5  5
  5  5  5  6  5  4  6  5  5  5  6  6  5  5  5  5  5  6  5  5  4  5  5  5
  5  5  5  6  5  5  6  6  5  5  6  5  5  5  5  5  5  6  5  4  5  5  5  6
  5  4  5  5  5  5  5  6  5  5  5  5  5  5  5  6  4  6  5  5  5  6  5  6
  5  4  5  5  5  6  5  5  5  5  5  4  5  5  5  5  4  5  5  6  6  5  6  6
  5  5  5  5  5  5  5  4  5  5  6  4  5  5  5  5  6  5  5  5  5  5  4  5
  6  5  5  6  6  5  6  5  5  5  7  6  6  5  6  5  5  5  5  5  5  5  5  5
  5  5  5  5  6  6  4  5  5  5  5  5  5  6  4  5  5  6  5  4]
```

Fig 3.40

Figure 3.41 is the display of y_test using print

```
In [ ]:  print(y_test)
```

```
[ 4  5  4  5  7  6  4  5  4  4  7  5  5  5  5  5  6  7  6  7  5  5  5  6
  7  5  6  7  5  5  4  4  6  5  5  5  5  8  4  4  7  6  3  7  6  6  5  5
  6  6  6  6  4  5  4  6  5  5  4  4  8  8  5  4  4  4  6  6  4  4  4  6
  5  7  4  5  6  6  7  5  7  6  3  5  7  5  4  6  6  7  4  4  5  5  5  5
  6  5  6  8  5  7  5  4  5  4  6  5  4  7  5  4  5  5  5  5  7  5  6  7
  5  6  5  4  4  5  4  6  4  5  5  5  5  5  4  5  5  5  6  6  4  5  4  6
  5  7  6  5  3  5  5  7  5  5  5  7  5  5  5  7  5  5  5  4  7  7  6  3
  5  5  5  5  6  4  5  6 10  4  6  6  4  7  5  5  4  5  4  6  4  3  5  5
  4  7  4  7  6  5  4  5  5  4  5  7  5  4  6  6  5  6  5  4  4  3  5  7
  6  6  4  6  6  3  4  4  4  5  7  6  5  6  4  6  3  6  6  4  2  2  4  7
  5  6  5  5  5  9  6  6  4  4  4  5  4  2  7  6  5  6  6  6  7  5  5  7
  5  4  6  6  4  5  5  5  6  6  5  4  2  4  4  6  8  4  6  5  4  6  6  5
  6  4  5  6  7  6  6  6  5  7  5  5  5  5  5  3  6  6  4  5  5  6  3  5
  7  5  6  4  6  5  5  4  6  5  3  5  5  4  5  6  6  4  3  3  7  5  6  6
  5  5  7  5  5  5  6  5  5  5  6  6  6  6  5  5  4  6  6  8  4  4  4  5
  5  5  6  7  5  4  7  4  5  5  6  5  5  6  6  4  6  4  5  5  5  4  6 10
  4  6  5  5  5  7  5  6  4  4  9  5  5  3  6  8  5  4  5  7  4  7  4  6
  3  5  7  5  5  6  3  6  5  7  3  5  6  7  6  5  4  5  5  4  5  6  7  7
  7  3  6  5  7  6  6  5  6  5  5  4  5  6  7  6  4  5  5  7  5  5  4  5
  5  5  4  7  6  6  4  4  4  5  7  5  6  5  6  5  6  5  4  3  6  6  6  6
  6  6  5  5  4  6  5  6  4  5  3  5  4  9  5  7  5  5  6  5]
```

Fig 3.41

Accuracy_score is a function from sklearn.metrics module which is used to compute the accuracy classification.The function takes y-test whhhh is actual target value and y_pred which is the target value as arguments and prints the value in the figure below. (Fig. 3.42)

```
In [ ]: from sklearn.metrics import accuracy_score

In [ ]: print(accuracy_score(y_test,y_pred)*100)

        39.6
```

Fig 3.42

Fig. 3.43 Prints the X_test values

```
In [ ]: print(X_test)

[[-0.78717838  0.31002559 -0.11522021 ... -0.08708126  0.02938985
  -0.06816691]
 [ 1.27036009  0.31002559 -0.11522021 ... -0.08708126 -0.87679731
  -1.32090317]
 [-0.78717838  0.31002559 -0.11522021 ... -0.60032122 -0.27267253
  -0.06816691]
 ...
 [ 1.27036009  0.31002559 -0.11522021 ... -0.60032122  1.84176419
   1.68566386]
 [-0.78717838 -3.2255402  -0.11522021 ... -2.39666109 -2.38710926
   0.4329276 ]
 [-0.78717838  0.31002559 -0.11522021 ... -2.39666109  0.63351463
   0.4329276 ]]
```

Fig 3.43

The code lodes the trained model which is saved in the filename variable using the method from pickle module.The predict method is used on the loaded model to make predictions on the test X_test and store the values and print the result.

```
In [ ]: # load the model from disk
        loaded_model = pickle.load(open(filename, 'rb'))
        result = loaded_model.predict(X_test)
        print(result);
```

Fig 3.44

Fig. 3.45 shows the results of predicted X_test. Datatime module is a bulit-in Python module that provides classes for working with dates and times. (Fig. 3.46) Datatime.date is used to represent the date for the thinitialal interface which can be changeable further in the UI. Then it prints the start_date addition with datetime.timedelta method which is in 10 days. We load the preprocessors into a dataset for binary classification. The dataset is divided into training and testing sets using StandardScalar features to scale. The model uses a pickle to save the data. The evaluation on the testing model is done and compute the score.

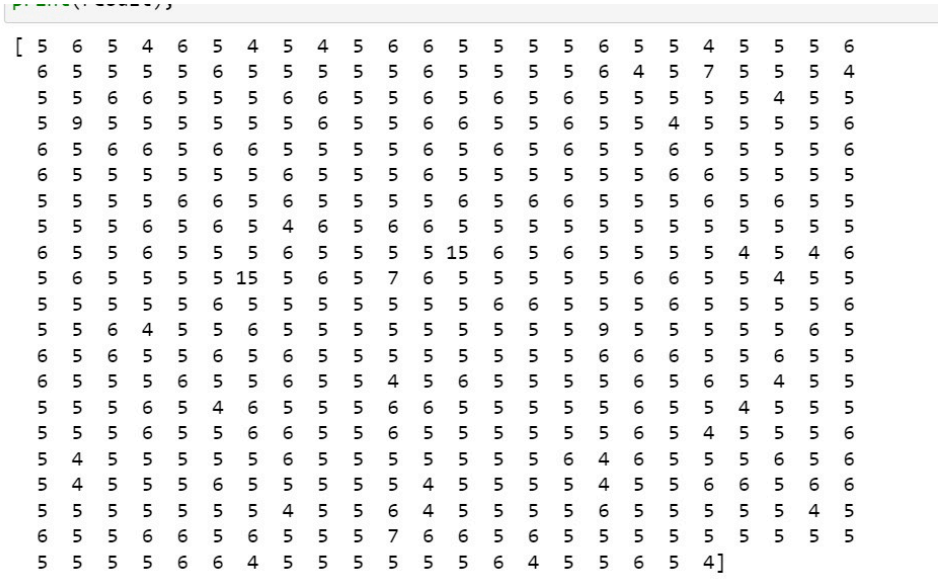


Fig 3.45

```
In [ ]: import datetime

In [ ]: start_date = datetime.date(2023, 4, 15);

In [ ]: print(start_date+datetime.timedelta(days=10));

        2023-04-25
```

Fig 3.46

3.5 List of tools used

The implementation of this project has been divided into 3 components:

- User interface
- Menstrual Cycle Prediction
- Breast Cancer Prediction

User Interface: To develop the calendar and other UI elements, we have used a special library called streamlit which is used for creating and sharing custom webpages for the machine learning algorithms.

Menstrual Cycle and Breast Cancer Prediction: This project is uses Logistic Regression for prediction. It has been developed using Python Language which is the easiest and most simple language. Python also has a variety of libraries, including matplotlib, NumPy, pandas, sklearn, etc., which makes it simple to create heatmaps or graphs, among other things. In this code, we have used libraries called Numpy, Matplotlib, Sci-Kit learn, and Pandas.We have used Jupyter Notebook which is open-source. It is so because showing the data visualizations and all other aspects of the project is easy using this tool.

Every tool used in our project is open source, which helps to lower project costs.

4. Project Management

4.1 Methodology used

As we are a small team of 5 members and everyone worked remotely during their own schedule, we chose to use agile methodology. Since the focus of this method is on collaboration and flexibility, we frequently communicated with each other through mails and messages along with a sync up post every class. Initially, the tasks were first divided among the teammates based on everyone's technical strengths and weaknesses. Although we haven't used any formal progress tracking tools, everyone ensured that they updated the status in our group chat so that the project progresses as planned.

We also implemented cross testing where the ones who didn't implement a particular component performed its testing. We made sure that each component was tested well and gave each other feedback on the developed features. This helped us in making the application better. Moving forward, we want to continue using agile methodology for the next increment of our project.

4.2 Work completed so far

4.2.1 Description

Following are the features that have been developed as a part of this increment:

- *User Interface:* We developed a user interface with features like login, calendar, predicting next cycles, adding period data etc.
- *Data Collection:* We collected data from multiple sources so that the models can be trained and tested. Also, we made sure that different age groups are considered in the data. For breast cancer prediction, we collected data from medical records and used other relevant parameters.
- *Menstrual Cycle Prediction:* We built a machine learning model that predicts when the user's next period will occur based on the previous months' data.
- *Breast Cancer Prediction:* We built a model to predict the user's risk of breast cancer based on various factors like age, family history etc.
- *Model Evaluation:* We also validated the accuracy of the models that have been built. We performed cross-validation to ensure that the models are not overfitting.

Overall, we have made a lot of progress in building this application. It helps women track their menstrual cycles and predict their risk of breast cancer. The UI and the models have been developed successfully, and we are now focusing on enhancing the UI, adding more features like notetaking, and including more health predictions in the app.

4.2.2 Responsibility and Contributions

Srividya Kamakshi Valiveti (20%)	• Work on the User interface for menstrual cycle tracking including the login, calendar and other aspects • Contribute towards building an ML model for breast cancer prediction. • Act as Scrum master to ensure project stays on track • Work on the final report for the increment – Overall formatting, Key aspects, roles and responsibilities, goals & objectives etc.
Sai Sree Raghavi Konda (20%)	• Find all datasets available on Breast cancer • Data preprocessing on the datasets to improve accuracy • Contribute towards building an ML model for breast cancer prediction • Create git repo so that the project can be maintained there • Documentation for how the model has been implemented
Charitha Sree Surineni (20%)	• Research on the data available to build an ML model for Menstrual tracking. • Contribute towards building an ML model to predict a user's menstrual cycle for the subsequent months. • Documentation for how the model has been implemented • Test all the components to ensure everything is okay
Bhavana Nalabothu (20%)	• Research on how ovulation, fertility tracking works. Gather all metrics that are needed medically for this project • Find all datasets that are relevant to the project • Currently implementing the notifications component • Work on the final report for the increment – Background work, List of tools to be used, workflow diagrams
Hemalatha Yella (20%)	• Worked on the UI calendar application for menstrual tracking and other symptoms like intensity etc. • Curate all the relevant resources that women will find useful in the application • Documentation for how the UI was implemented, what all technologies were used • Test the system to find any bugs or gaps

In any software team, members take up different roles based on their skills. But for the project to be a success, everyone must contribute equally. If not, it will surely fail. Since everyone in our team worked together, we felt that all of us have contributed equally to all areas of the project.

4.3 Work to be completed

Although we have made quite a lot of progress in the project implementation, we have some work that is left to do as a part of the next increment. Following are the high-level details of features yet to be implemented:

- *Enhancing UI:* The current user interface of our application is basic and lacks visual appeal. We want to modify it using a better design so that more users can utilize the application.
  (Currently being implemented)
- *Integrating Breast Cancer prediction:* The model which is predicting breast cancer has been developed separately. We plan to integrate it into the UI for easy access.
  (Currently being implemented)
- *Notes to a date:* This is an exciting feature that we want to take up where users can add notes to a specific date. This will allow them to keep track of their moods, symptoms experienced on that date like body pains or any other important information which might be useful later.
- *Notifications:* A notification feature will help in reminding the user about their upcoming menstrual cycle or any notes they have added for a particular date. It will remind them of these so they can use it wherever needed.
  (Currently being implemented)
- *Final testing and issue fixes:* We want to perform thorough testing of the whole application so that if there are any issues, we will resolve them and refine the application.

5. Conclusion

In conclusion, our project has met its primary objective of being a one-stop solution for women to track their menstrual cycle, predict future periods and screen themselves for the risk of having breast cancer. Using Agile methodology has allowed us to work efficiently while being remote, ensuring that the project got completed on time.

The application can be even more useful for women and could be a valuable tool for their health if we add other features for health situations like Menopause, PCOD/PCOS, Endometriosis etc. Making the user interface better and integrating breast cancer model into it will really make the UI better. It will provide a seamless experience to all the users while using this application. This will lead to more and more woman becoming regular users of this application.

Overall, Sakhi, Every Women's Health Care Assistant is an essential tool for women, providing them with the ability to monitor their menstrual cycle, track their health, and identify potential health concerns. This application can have a positive impact women's health, and with some more improvements, it will become a vital resource for women of all ages one day.

6. References

[1] Boivin J, Bunting L, Collins JA, et al. International estimates of infertility prevalence and treatment-seeking: potential need and demand for infertility medical care. Human reproduction (Oxford, England) 2007;22(6):1506-12. doi: 10.1093/humrep/dem046 [published Online First: 2007/03/23]

[2] "Menstrual Cycle and Fertility App Development: A Systematic Review" by Asghari et al. This article reviews existing menstrual cycle and fertility apps and provides insights into the features that are most useful to users.

[3] World Health Organization's report on rise in Infertility

[4] How common is Breast Cancer? American Cancer Society

[5] Breast Cancer Prediction Using Machine Learning: A Review" by Nourhene Farhat and Chafik Aloulou. Link: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7574947/

[6] National Breast Cancer Foundation: Breast Self-Exam
https://www.nationalbreastcancer.org/breast-self-exam

[7] "A Comprehensive Review of Applications for Period Tracking" by Lim et al. This article provides an overview of the features and functionalities of various period tracking apps and compares their accuracy and usability.

[8] Rutstein SO, Shah IH. Infecundity infertility and childlessness in developing countries. Geneva: World Health Organization 2004.

[9] National Cancer Institute. (2021). Breast cancer prevention (PDQ®)–patient version. Retrieved from https://www.cancer.gov/types/breast/patient/breast-prevention-pdq

[10] Hosmer, D. W., Jr., Lemeshow, S., & Sturdivant, R. X. (2013). Applied logistic regression. John Wiley & Sons.
This is a comprehensive and widely used textbook on logistic regression. It covers topics such as model building, model diagnostics, and model validation.

[11] The pandas documentation: https://pandas.pydata.org/docs/. This is the official documentation for pandas, which includes a comprehensive user guide, API reference, and tutorials. It is the best place to start for learning pandas.

[12] The Streamlit documentation: https://docs.streamlit.io/en/stable/. This is the official documentation for Streamlit, which includes a getting started guide, API reference, and examples. It is the best place to start for learning Streamlit.