DETECTION OF POLYCYSTIC OVARY SYNDROME (PCOS) BASED ON PATIENT DATA

A PROJECT REPORT

Submitted By

Anuradha Nair, Roll No- 12621018011, Reg No-211260130810006
S Prabha, Roll No- 12622018072, Reg No- 221260121008
Srimoyee Sarkar, Roll No- 12621018049, Reg No- 211260130810038
Ritwika Bera, Roll No- 12621018036, Reg No- 211260130810051

Under the supervision of

Prof.
(Dr.)
Debamita Kumar
Assistant
Professor
Dept. of Computer Science and Engineering(AIML)
Heritage Institute of Technology, Kolkata

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

HERITAGE INSTITUTE OF TECHNOLOGY, KOLKATA

An autonomous Institute under
Maulana Abdul Kalam Azad University of Technology
Formerly Known as a
West Bengal University of Technology

July, 2024

ACKNOWLEDGEMENT

We would take this opportunity to thank Prof. (Dr.) Pranay Chaudhuri, Principal Heritage Institute of Technology for providing us with all the necessary facilities to make our project work successful.

We would like to thank our Head of the Department Prof. (Dr.) Sujay Saha for his kind assistance as and when required.

We will be thankful to Prof. (Dr.) Debamita Kumar our project coordinator for constantly supporting and guiding us for giving us invaluable insights. Her guidance and her words of encouragement motivated us to achieve our goal and impetus to excel.

We thank our Faculty members and Laboratory assistants at the Heritage Institute of Technology for paying a pivotal and decisive role during the development of the project. Last but not the least we thank all friends for their cooperation and encouragement that they bestowed on us.

(Anuradha Nair)

(S Prabha)

(Srimoyee Sarkar)

(Ritwika Bera)

HERITAGE INSTITUTE OF TECHNOLOGY, KOLKATA

An autonomous Institute under

Maulana Abdul Kalam Azad University of
Technology Formerly Known as a
West Bengal University of Technology

BONAFIDE CERTIFICATE

Certified that this project report DETECTION OF POLYCYSTIC OVARY SYNDROME(PCOS) BASED ON PATIENT DATA is the bonafide work Anuradha Nair, S Prabha, Srimoyee Sarkar and Ritwika Bera who carried out the project work under my supervision.

SIGNATURE

SUJAY SAHA

HEAD OF THE DEPARTMENT

COMPUTER SCIENCE AND ENGINEERING(AIML)

SIGNATURE

DEBAMITA KUMAR

PROJECT GUIDE

ASSISTANT PROFESSOR

COMPUTER SCIENCE AND ENGINEERING(AIML)

SIGNATURE

EXAMINER

ABSTRACT

Polycystic Ovary Syndrome (PCOS) is a common hormonal disorder affecting women of reproductive age, characterized by irregular periods, excess androgen levels, and polycystic ovaries. It can lead to complications like infertility, diabetes, and cardiovascular issues if left undiagnosed or untreated.

Developing a PCOS detection system using machine learning (ML) can aid in early diagnosis by analyzing complex patterns in medical data, improving accuracy, and allowing for timely intervention, thus enhancing patient outcomes and reducing long-term health risk.

By utilizing machine learning models like Logistic Regression, Support Vector Machines (SVM), and Random Forest, the PCOS detection system can be trained to analyze various medical and clinical features such as hormonal levels, body measurements, and lifestyle factors. These algorithms process the data to identify hidden patterns and relationships that may not be immediately apparent through traditional diagnostic methods. The integration of such systems with healthcare not only enhances diagnostic precision but also reduces the dependency on invasive procedures.

Feature engineering plays a pivotal role in improving the model's performance. Selecting the most relevant features, such as fasting insulin levels, menstrual cycle regularity, and ultrasound findings, ensures that the model focuses on critical indicators of PCOS. Additionally, preprocessing techniques like normalization or scaling further refine the data, enabling algorithms to process it efficiently and reduce bias.

Incorporating machine learning in PCOS detection marks a significant advancement in women's health. As models evolve and datasets expand, the accuracy and applicability of such systems will only improve, making them an indispensable tool in modern medicine. The integration of these technologies could bridge gaps in accessibility, enabling even remote or underserved regions to benefit from timely and accurate PCOS diagnosis.

TABLE OF CONTENTS

1. INTRODUCTION

Polycystic Ovary Syndrome (PCOS) is a significant and growing global health concern, affecting approximately 6-12% of women of reproductive age, according to recent estimates. It is one of the leading causes of infertility and is linked to serious long-term health issues such as type 2 diabetes, obesity, cardiovascular diseases, and even mental health disorders like anxiety and depression. Despite its prevalence, PCOS remains underdiagnosed, with many women experiencing delayed treatment due to its complex and variable symptoms.

One of the major concerns is infertility, as PCOS often leads to irregular or absent ovulation, making it difficult for women to conceive. Even when pregnancy is achieved, women with PCOS are at a higher risk of miscarriage, gestational diabetes, preeclampsia, and preterm birth, all of which can complicate pregnancy and pose risks to both mother and baby.

Common symptoms of PCOS, such as excessive hair growth (hirsutism), acne, and weight gain, are linked to hormonal imbalances and can cause emotional distress.

With rising cases worldwide and its profound impact on quality of life, addressing PCOS through improved detection and personalized treatment is critical, making the role of advanced technologies like machine learning (ML) vital in tackling these challenges efficiently.

Machine learning offers a transformative approach to PCOS detection by analyzing vast and complex medical datasets to identify patterns and correlations that traditional methods might overlook. Models like Logistic Regression, Support Vector Machines, and Random Forests are particularly effective in handling diverse features such as hormone levels, lifestyle factors, and ultrasound findings, enabling more accurate and faster diagnosis.

Incorporating machine learning not only enhances diagnostic precision but also facilitates early intervention, which is essential in mitigating long-term health risks associated with PCOS. Personalized predictions based on patient-specific data allow healthcare providers to develop tailored treatment plans, addressing individual needs and improving overall outcomes.

2. LITERATURE SURVEY

• In the Research Paper that we have added in the Reference Section, over there they have achieved a higher accuracy compared to ours. This is because they have used Bayesian Optimization with cross-validation to optimize ML algorithms and enhance accuracy.

• The Research Paper has used complex architectures, high computation methods which is time consuming, but we have achieved an Accuracy nearer to the one which the research work has achieved by using simpler methods within less time frame.

• Despite not utilizing advanced techniques like Bayesian Optimization, our approach demonstrates that simpler methods can achieve competitive performance, making them more accessible and practical for real-world applications.

• Our model's faster execution time and reduced computational requirements make it a cost-effective solution for healthcare settings, especially in resource-limited environments.

• By focusing on straightforward preprocessing and standard machine learning models, we have prioritized ease of implementation without significantly compromising accuracy.

Advantages of Our Approach:

- **Efficiency:** Lower computational complexity ensures quick model deployment.
- **Accessibility:** Simpler techniques make the approach usable in regions with limited technical resources.
- **Scalability:** The method can be easily adapted or extended to other medical datasets.
- **Practicality:** Provides a reliable solution for PCOS diagnosis without requiring high-end infrastructure.

3. METHODOLOGY

The project involves fetching data from Kaggle to analyze PCOS-related datasets: *PCOS_without_infertility* (541 samples, 45 features) and *PCOS_with_infertility* (541 samples, 6 features), comprising 177 positive and 364 negative cases. Using libraries like Pandas, NumPy, Matplotlib, Seaborn, and Scikit-learn, preprocessing steps included merging the datasets, separating numerical and categorical features, encoding with `pd.to_numeric`, and handling null values.

Exploratory Data Analysis (EDA) involved creating t-SNE plots, analyzing feature correlations with the target variable (e.g., Follicle No, Skin Darkening, Hair Growth), identifying irrelevant features (e.g., FSH, FSH/LH), and visualizing data via histograms and pair plots.

For modeling, a 70-30 train-test split was applied, with features normalized using StandardScaler. Models tested included Logistic Regression, Random Forest, and SVM classifiers.

3.1 PRE-PROCESSING

Merging the two files PCOS_with_infertility and PCOS_without_infertility , Separating numerical and categorical features, Encoding Categorical features- pd.to_numeric, Handling of NULL values.

Merging the datasets *PCOS_with_infertility* and *PCOS_without_infertility* allows for a comprehensive analysis by combining all relevant data into a single framework, facilitating better insights into patterns and trends.

After merging the **Pandas Profiling** library is used to generate a detailed and interactive exploratory data analysis (EDA) report for the combined DataFrame. This report is displayed as an iframe in a Jupyter Notebook and saved as an HTML file (e.g., "PCOS_dataset_profiling_report.html") for offline use.

After the Pandas Profiling, the separation of numerical and categorical features is essential for targeted preprocessing. Numerical features can be scaled and normalized to improve model performance, while categorical features are encoded using pd.to_numeric or similar encoding techniques, ensuring that machine learning algorithms can interpret them effectively.

Handling null values is a critical step to maintain the integrity of the dataset. Missing values can be addressed through imputation methods such as mean or median for numerical features and mode for categorical features. In cases of excessive missing data, rows or columns may be removed if they contribute minimally to the analysis.

3.1.1 Missing Data

Missing data is a common problem and one of many factors comprising data quality. Data quality of a dataset depends on the intended use of the data, meaning a dataset may be of good quality for one use case, but not for another. Missing data or missing values are values containing false data, or no value at all. A value in an object might be missing, or even the whole object. Sometimes, whole records are missing from a dataset. Having values that are missing in a dataset may affect the quality of the data, and therefore affect information to be gathered from the dataset. Missing data is a common problem within industrial and medical databases, and is a source of serious problems in statistical analyses of clinical trials and of other medical studies. It is not uncommon to find datasets containing up to 50% missing values. Common ways to handle missing data is to ignore them, or remove the cases containing missing values, commonly known as case deletion. However, in some cases, inferring the values with an imputation method is necessary. Finding the right approach to handle the missing data depends on the application domain and the data itself. Therefore, understanding the data and its type is important when handling missing data. The values might be missing because of random errors with equipment or calculation, attrition due to social or natural processes as for instance death, or respondent refusal, such as a person not answering certain questions in a survey. Some data might even be intentionally missing.

Types of Missing Data:

1. Missing at Random (MAR): Missing at random means that the propensity for a data point to be missing is not related to the missing data, but it is related to some of the observed data
2. Missing Completely at Random (MCAR): The fact that a certain value is missing has nothing to do with its hypothetical value and with the values of other variables.
3. Missing Not At Random (MNAR): Two possible reasons are that the missing value depends on the hypothetical value (e.g. People with high salaries generally do not want to reveal their incomes in surveys) or missing value is dependent on some other variable's value (e.g. Let's assume that females generally don't want to reveal their ages, here the missing value in age variable is impacted by gender variable)

In the first two cases, it is safe to remove the data with missing values depending upon their occurrences, while in the third case removing observations with missing values can produce a bias in the model. Hence, we have to be really careful before removing observations. Note that imputation does not necessarily give better results.

3.1.2 Handling Missing Data

The provided code snippet demonstrates handling missing values in a dataset. It begins by identifying columns with missing (NA) values using the Pandas method `isna()`, which checks for missing values in the dataset. The `any()` function ensures that only columns with at least one missing value are selected, and these column names are stored in the list `columns_with_na`, which is then printed for the user's reference. Following this identification, the missing values in specific columns are handled by filling them with the median of the respective feature. The median is chosen because it is a robust measure of central tendency that is less affected by outliers compared to the mean, making it suitable for numerical data that may have extreme values. For example, the code fills missing values in the `Marraige Status (Yrs)` column with its median, ensuring that the imputed value represents the central trend of the existing data without being skewed by outliers.

Similarly, missing values in the `II beta-HCG(mIU/mL)` and `AMH(ng/mL)` columns, which appear to be numeric biological measurements, are also filled with their respective medians. For the column `Fast food (Y/N)`, the same approach is applied. Although this column seems to be categorical, it likely contains binary numeric representations (e.g., 1 for Yes, 0 for No), allowing the median imputation method to work. This approach ensures that the dataset remains complete and ready for further analysis or modeling while minimizing the risk of introducing bias or inconsistencies caused by missing values.

3.2 FEATURE SELECTION

`Feature Selection (FS) process is used for decreasing the dimensionality of the data and increasing the efficiency of learning algorithms. For the FS process, the whole search space covers all possible subsets of features and the number of subsets is calculated as given by the following equation:` $\sum \binom{n}{s}_{s=0} = \binom{n}{o} + \binom{n}{1} + \cdots + \binom{n}{n}$ (1) where n represents the number of features and s is the size of the current feature subset.

Feature selection is done by following a comprehensive approach to exploratory data analysis (EDA) and feature selection for a dataset related to PCOS diagnosis. It begins with t-SNE, a dimensionality reduction technique, applied to visualize the data in a 2D space. Initially, the features are selected by excluding non-relevant columns such as `Patient File No.` and the target variable `PCOS (Y/N)`. The selected features are then normalized using `StandardScaler` to ensure uniform scaling. The t-SNE algorithm is applied with two components, resulting in a 2D representation of the data. The resulting components are added as new columns, `tsne-2d-one` and `tsne-2d-two`, and visualized using a scatter plot. This plot categorizes data points into PCOS Positive and Negative classes. The plot reveals mixed data points, indicating limited separation in the 2D space, suggesting that the features may not fully distinguish between the two classes in this representation. Despite this, subtle patterns or clusters may exist, which require further analysis.

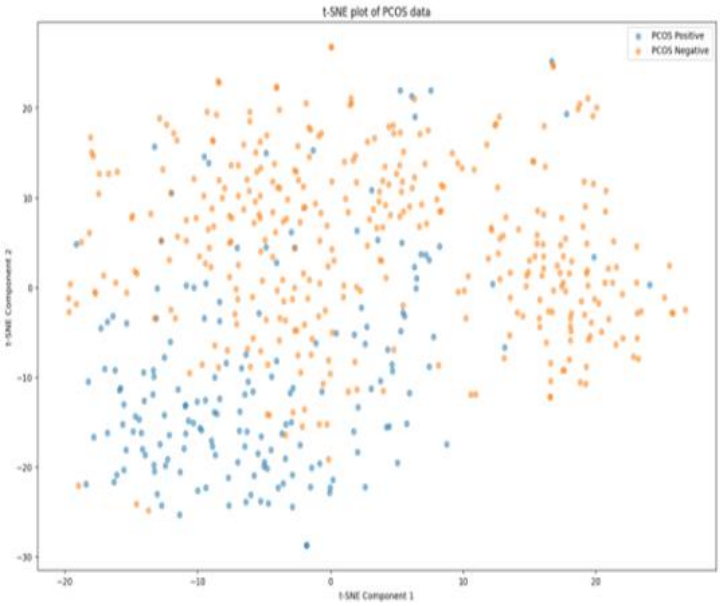
The next step involves examining the correlation between features and the target variable, `PCOS (Y/N)`. A correlation matrix is computed, and features are sorted based on their correlation values. A bar plot visualizes these correlations, identifying highly correlated features like `Follicle No. (R)`, `Follicle No. (L)`, `Skin Darkening (Y/N)`, `Hair Growth (Y/N)`, and `Weight Gain (Y/N)` as important predictors of PCOS. These features show strong relationships with the target variable, making them significant for further analysis or modeling. Conversely, less correlated features such as `FSH(mIU/mL)`, `FSH/LH`, and `PRG(ng/mL)` are identified as less important, given their minimal variation with the target variable. These findings are supported by pair plots, which visually explore the relationships between features and help distinguish important predictors from irrelevant ones.

To deepen the understanding of feature distributions, the dataset is visualized using histograms for selected

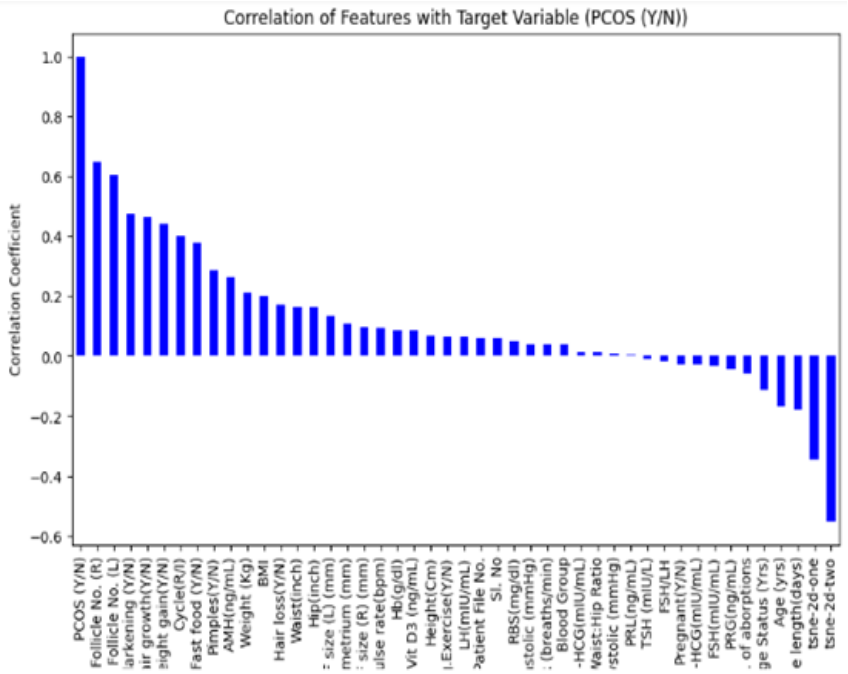
features. Separate histograms are plotted for PCOS Positive and Negative cases, highlighting the distribution differences. Features like `Follicle No. (R)`, `Follicle No. (L)`, `Skin Darkening (Y/N)`, and others demonstrate distinct density curves for the two classes, affirming their importance. In contrast, unimportant features such as `FSH(mIU/mL)`, `FSH/LH`, and `PRG(ng/mL)` show overlapping distributions, indicating limited discriminatory power.

This EDA process emphasizes key techniques for feature evaluation, including t-SNE plots for visual clustering, correlation analysis for statistical relevance, pair plots for relationship insights, and histograms for distribution comparisons. The combined approach ensures a thorough identification of important features for predicting PCOS, while also flagging irrelevant ones, setting a robust foundation for subsequent modeling. This analysis highlights the complex interplay between features and their impact on the target variable, aiding in developing accurate and interpretable machine learning models for PCOS classification.

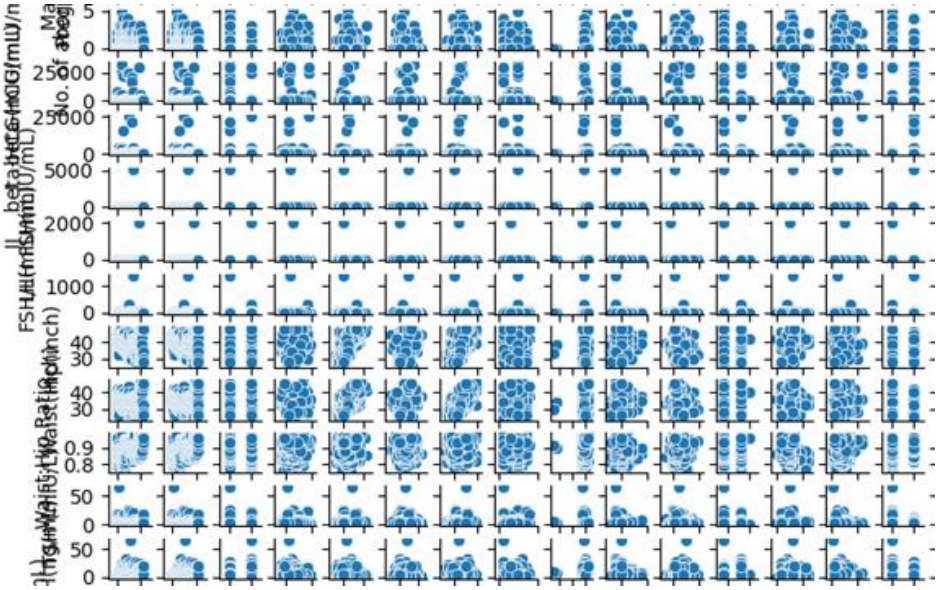
TSNE Plot



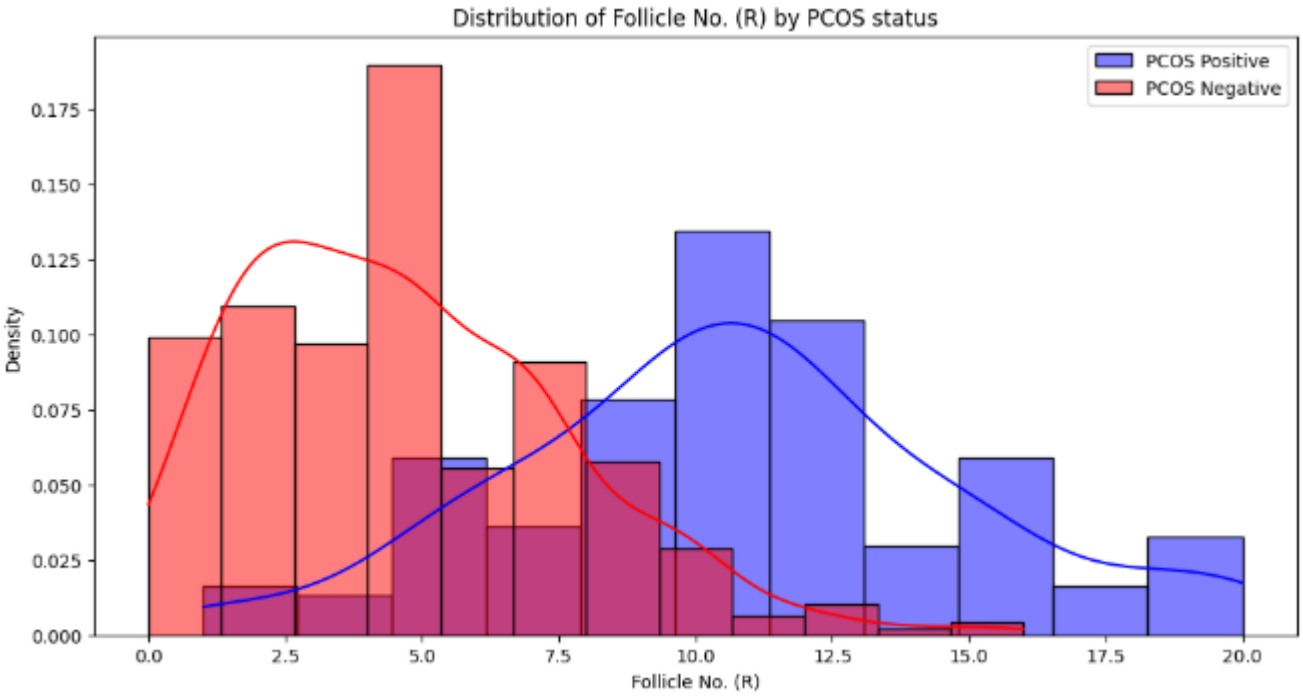
Correlation Plot

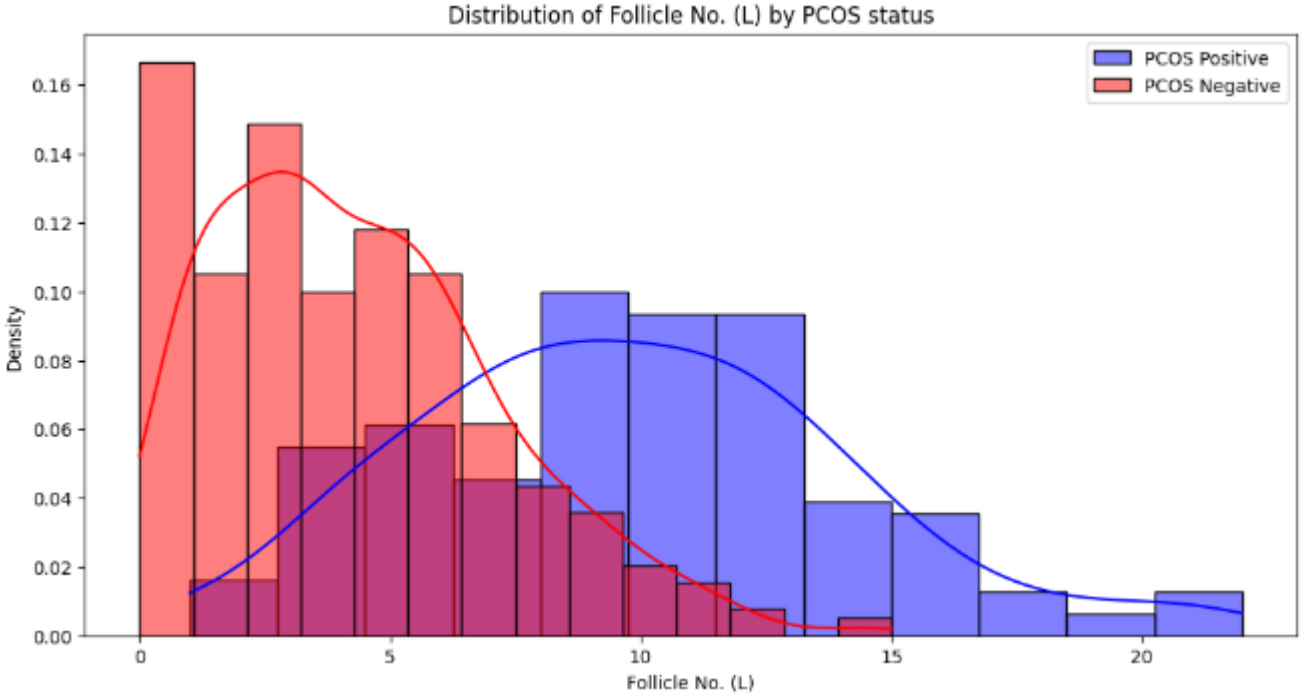


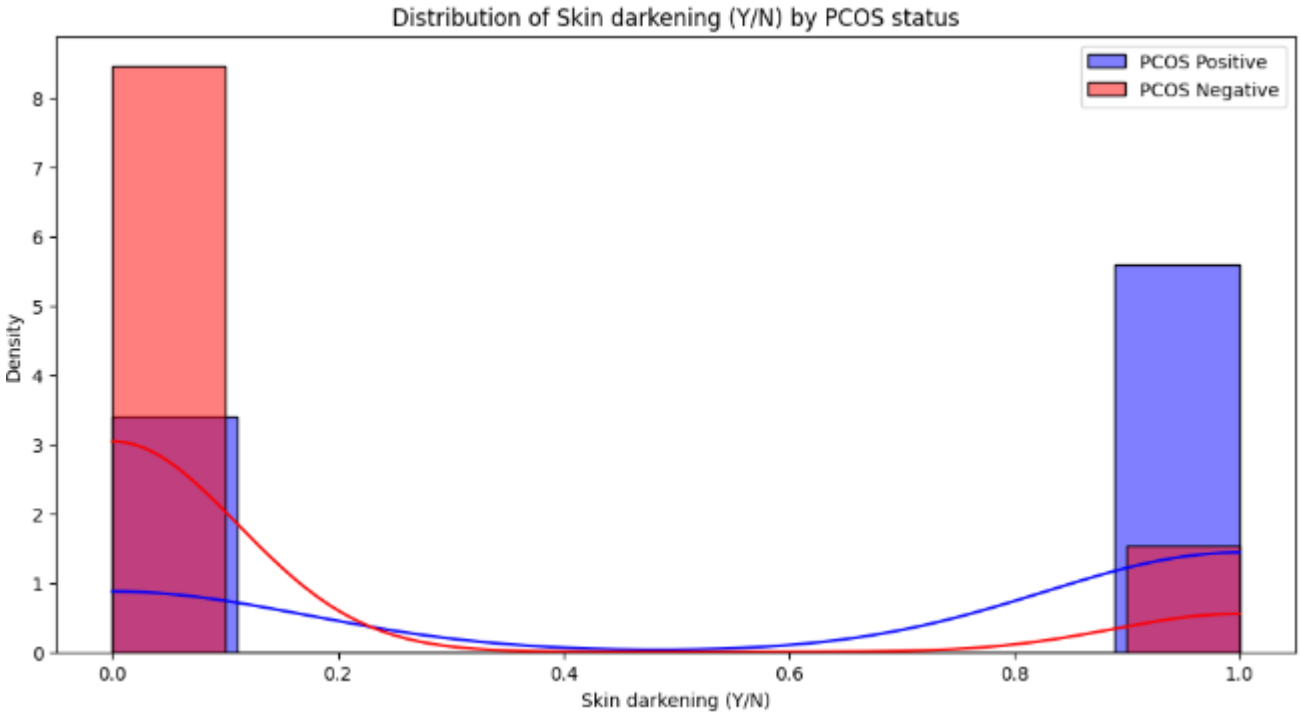
SNS Pairplot

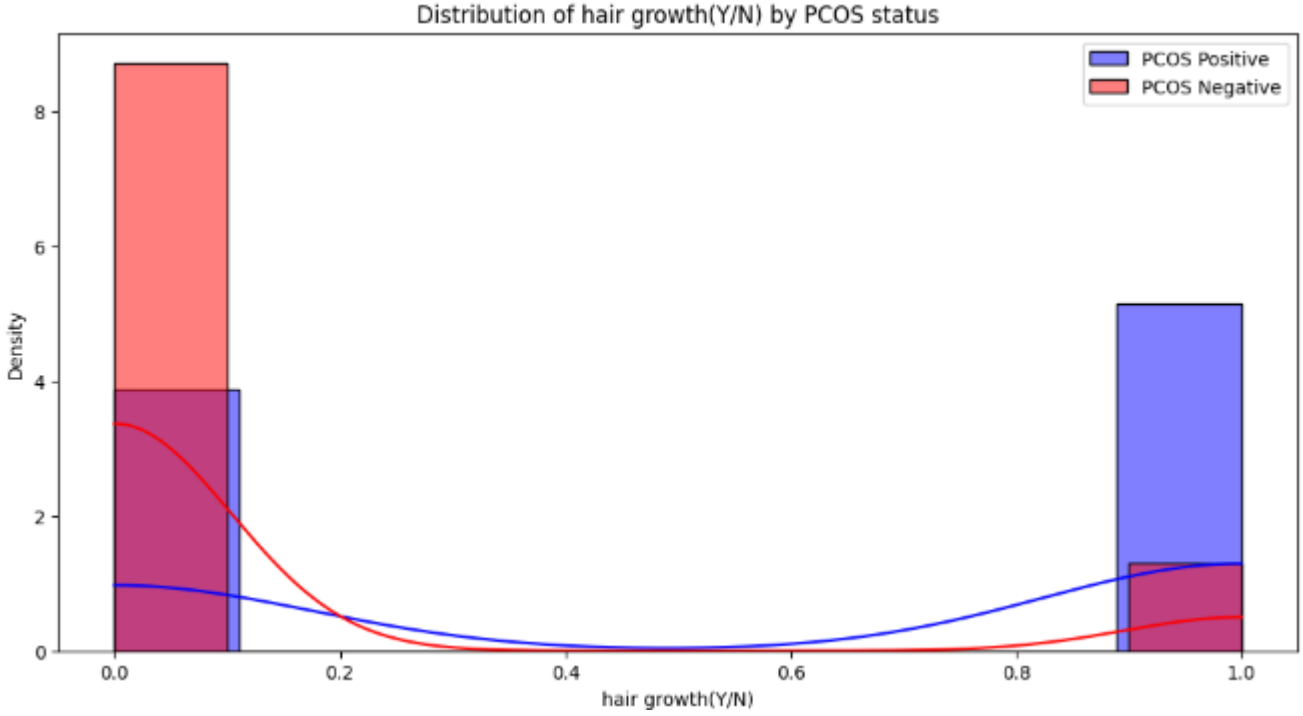


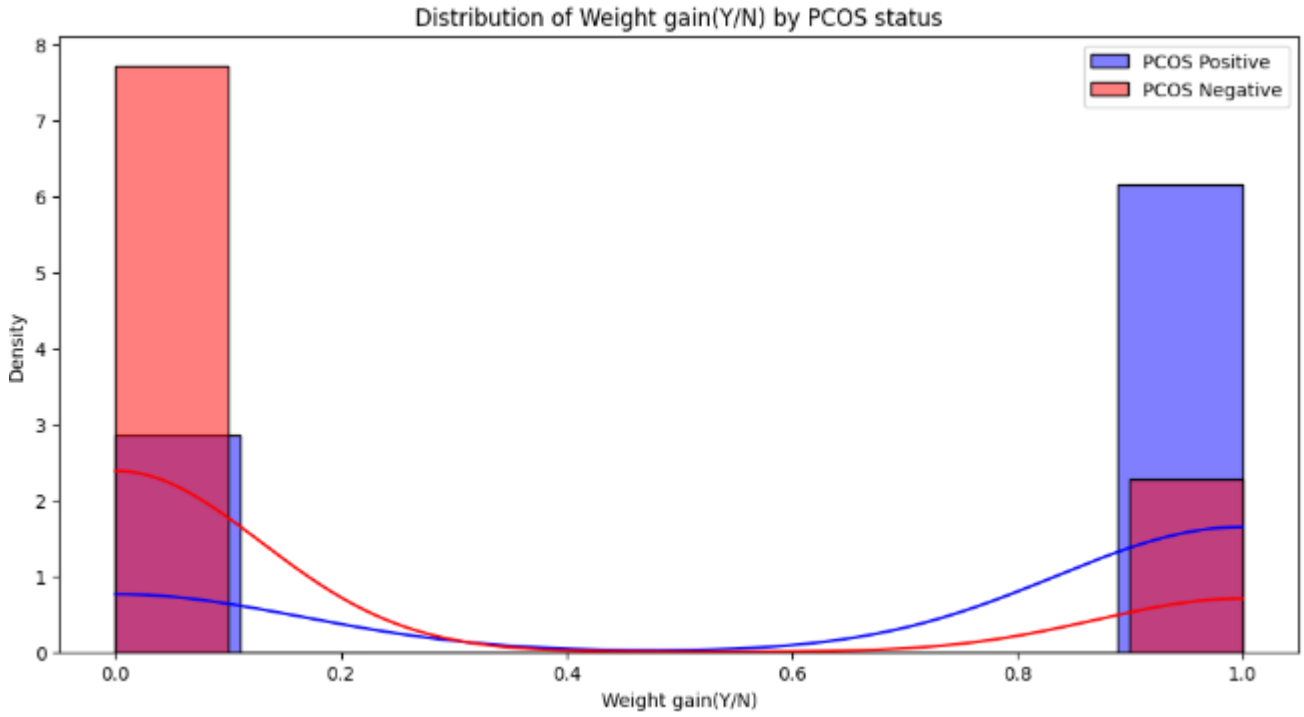
Histograms

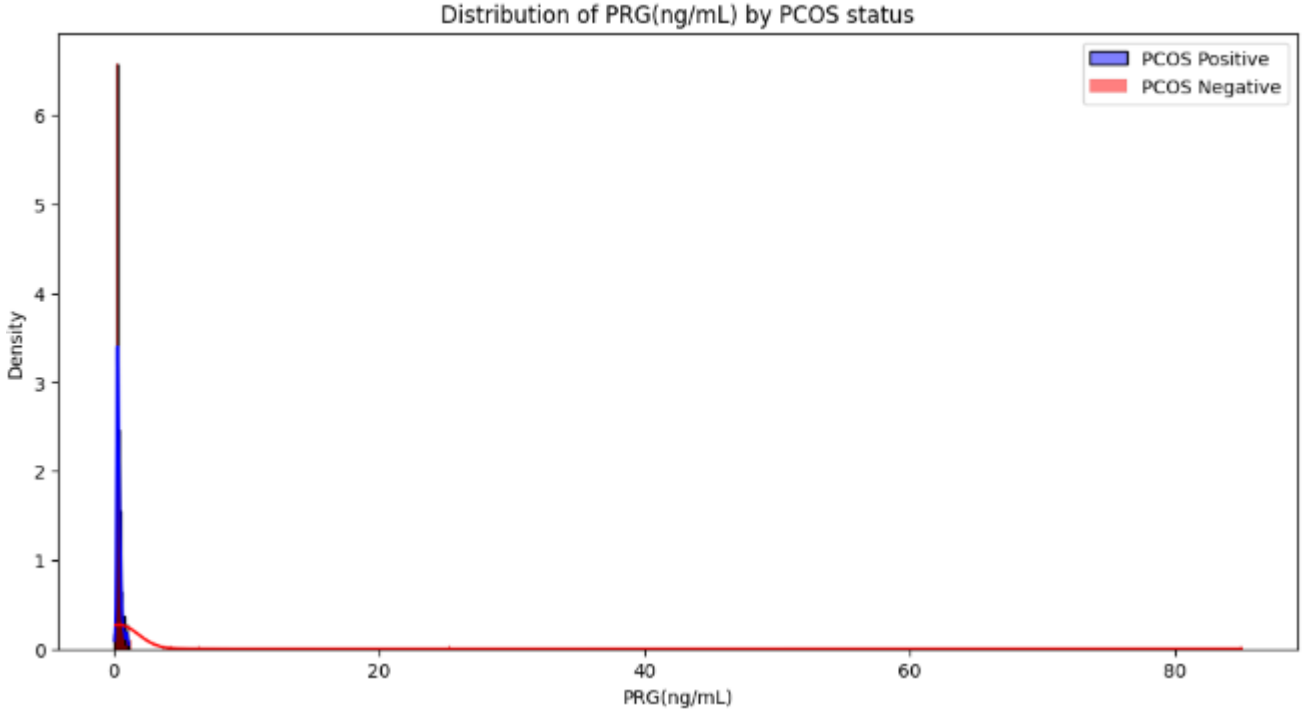

Distribution of Follicle No. (R) by PCOS status
PCOS Positive
PCOS Negative
Density
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Follicle No. (R)

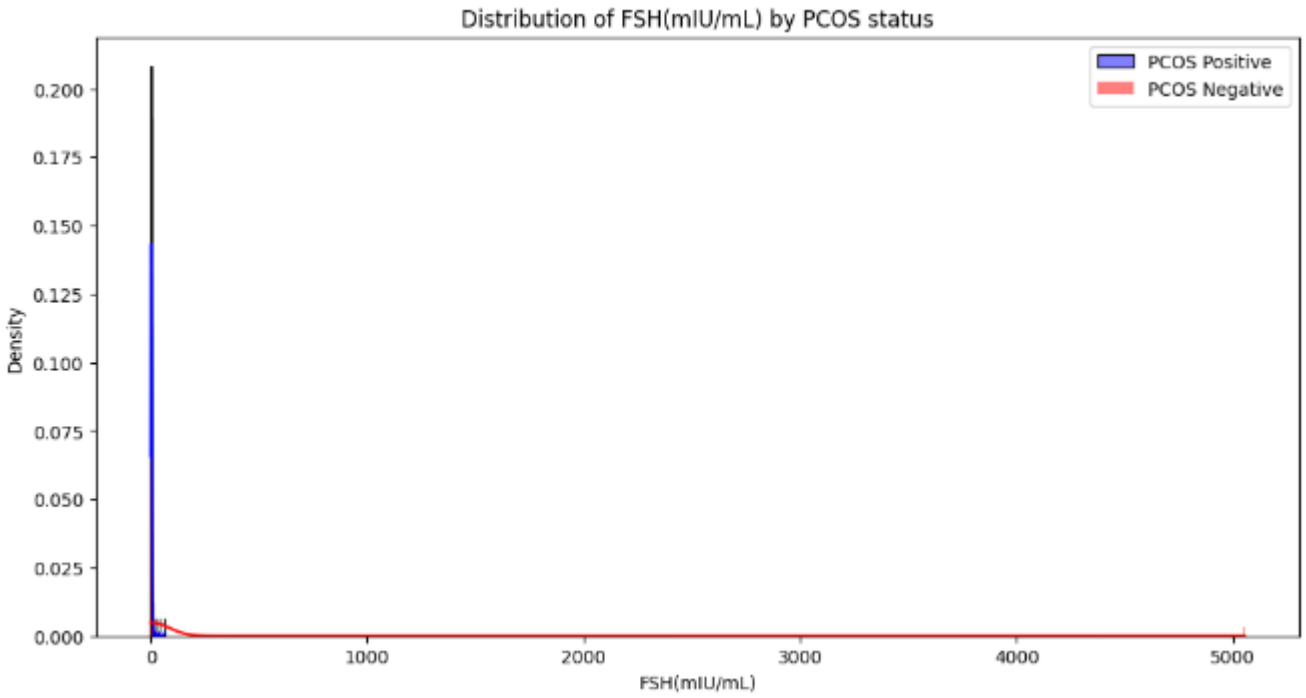

Distribution of Follicle No. (L) by PCOS status
PCOS Positive
PCOS Negative
Density
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
5
10
15
20
Follicle No. (L)

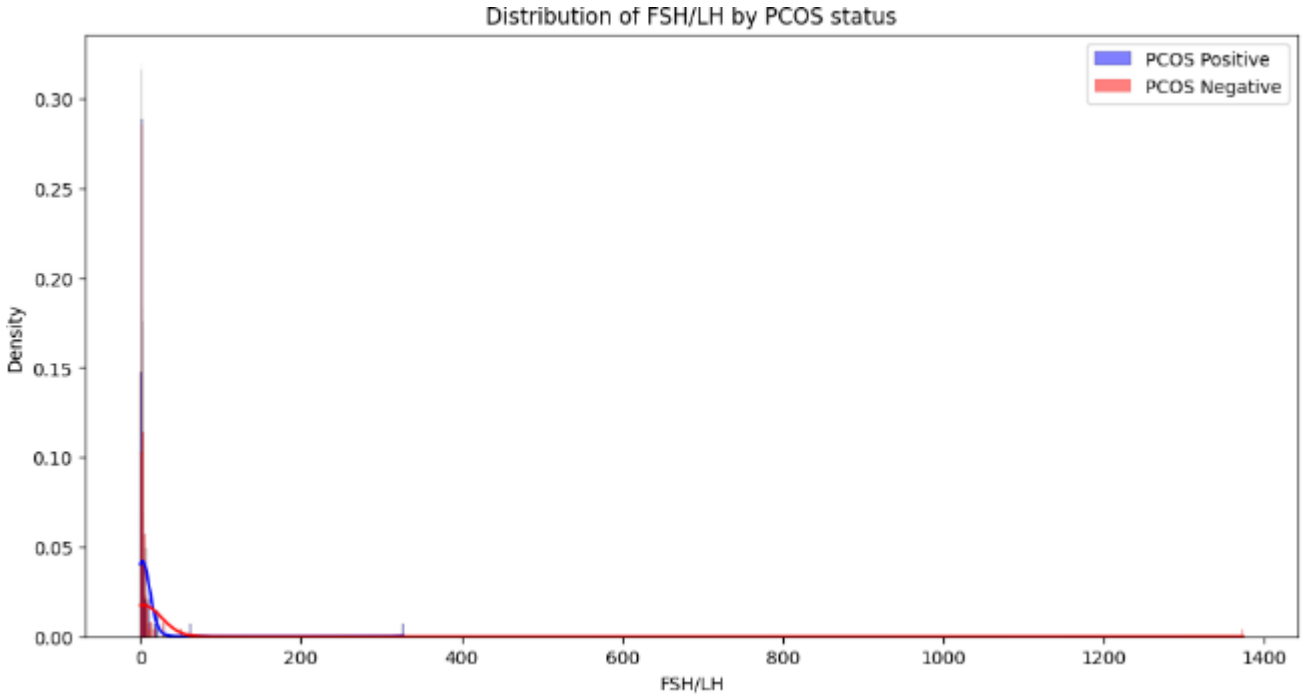

Distribution of Skin darkening (Y/N) by PCOS status
PCOS Positive
PCOS Negative
Density
Skin darkening (Y/N)
0
1
2
3
4
5
6
7
8
0.0
0.2
0.4
0.6
0.8
1.0


Distribution of hair growth(Y/N) by PCOS status
PCOS Positive
PCOS Negative
Density
hair growth(Y/N)
0
2
4
6
8
0.0
0.2
0.4
0.6
0.8
1.0


Distribution of Weight gain(Y/N) by PCOS status
PCOS Positive
PCOS Negative
Density
Weight gain(Y/N)
0
1
2
3
4
5
6
7
8
0.0
0.2
0.4
0.6
0.8
1.0


Distribution of PRG(ng/mL) by PCOS status
PCOS Positive
PCOS Negative
Density
PRG(ng/mL)
0
1
2
3
4
5
6
0
20
40
60
80


Distribution of FSH(mIU/mL) by PCOS status
PCOS Positive
PCOS Negative
Density
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
0
1000
2000
3000
4000
5000
FSH(mIU/mL)


Distribution of FSH/LH by PCOS status
PCOS Positive
PCOS Negative
Density
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
200
400
600
800
1000
1200
1400
FSH/LH

4. APPLICATON OF MODELS

4.1 Logistic Regression

Logistic Regression is a widely used supervised machine learning algorithm for binary and multiclass classification problems. It models the probability that a given input instance belongs to a particular class by applying a logistic function (also known as the sigmoid function) to a linear combination of the input features. Unlike linear regression, which predicts continuous values, logistic regression outputs probabilities ranging between 0 and 1. These probabilities are then thresholded (commonly at 0.5) to classify the instances into discrete categories. The logistic function ensures that the output is bounded and interpretable as a probability, making logistic regression especially effective for classification tasks. The algorithm works by optimizing a cost function called the log-loss or binary cross-entropy, which measures the error in predicted probabilities against the actual labels. This optimization process uses techniques such as gradient descent to iteratively adjust the weights of the features, aiming to minimize the cost function and improve prediction accuracy.

The full code demonstrates the practical application of logistic regression for classifying whether a patient has PCOS (Polycystic Ovary Syndrome) based on certain features. The data is split into training and testing sets to ensure a robust evaluation of the model's performance on unseen data. Before training the model, the features are normalized using the `StandardScaler` to ensure they have a mean of zero and a standard deviation of one. This step is crucial because logistic regression, being a linear model, is sensitive to the scale of input features, and normalization prevents any single feature from disproportionately influencing the model's performance.

The logistic regression model is then trained on the scaled training data. During training, it computes a decision boundary that separates the two classes (PCOS and non-PCOS) by maximizing the likelihood of the observed data given the logistic regression model. Once trained, the model is used to predict the class labels for the test data and also compute the probability of each instance belonging to the positive class. These probability estimates are particularly useful for calculating performance metrics such as the AUC-ROC score, which measures the model's ability to discriminate between the two classes across different decision thresholds.

The evaluation process involves calculating accuracy, precision, recall, and the F1 score. Accuracy provides an overall measure of the model's performance by comparing the number of correct predictions to the total number of predictions. Precision measures the proportion of correctly predicted positive cases out of all predicted positives, reflecting the model's ability to avoid false positives. Recall quantifies the proportion of actual positive cases that are correctly identified, indicating how well the model captures true positives. The F1 score, a harmonic mean of precision and recall, provides a balanced metric that accounts for both false positives and false negatives, making it particularly useful for imbalanced datasets.

A confusion matrix is generated to visually assess the model's performance. It summarizes the number of true positive, true negative, false positive, and false negative predictions, providing insights into specific types of errors made by the model. The confusion matrix is then plotted as a heatmap with clear annotations for easy interpretation.

The classification report complements the confusion matrix by presenting precision, recall, F1 score, and support for each class, offering a detailed view of the model's performance across all metrics.

Additionally, the ROC curve is plotted to illustrate the trade-off between the true positive rate (sensitivity) and false positive rate at various thresholds. The area under the curve (AUC) quantifies the overall performance of the model, with a higher AUC indicating better discrimination between the classes. The ROC curve and AUC score provide a robust evaluation metric independent of the specific threshold used for classification, making them valuable for comparing different models.

4.2 Random Forest

Random Forest is a robust ensemble learning method primarily used for classification and regression tasks. It builds a collection of decision trees during training, where each tree operates on a random subset of features and data samples. The predictions from all trees are aggregated (majority voting for classification and averaging for regression) to make the final prediction, ensuring reduced variance and overfitting. This algorithm leverages the concept of *bagging* (Bootstrap Aggregating) and feature randomness to enhance predictive accuracy and robustness. By combining the outcomes of multiple weak learners (decision trees), Random Forest achieves high performance, especially on complex datasets, by capturing both linear and nonlinear relationships.

The provided code trains and evaluates a Random Forest classifier on a dataset aimed at predicting PCOS diagnosis. The model is initialized with a fixed random state for reproducibility. The training process begins by fitting the model on the scaled training data (`X_train_scaled` and `y_train`), where feature scaling ensures uniformity in feature magnitudes for better performance. Predictions are then made on the test data (`X_test_scaled`), generating both discrete class labels (`y_pred`) and class probabilities (`y_pred_proba`). These probability estimates are critical for evaluating the model's performance through metrics like ROC-AUC.

To assess the classifier's accuracy, the `accuracy_score` is computed, reflecting the proportion of correct predictions. The confusion matrix provides detailed insights into the model's classification errors, illustrating the number of true positives, true negatives, false positives, and false negatives. This is visualized as a heatmap using Seaborn for clarity, with clear labeling of predicted and actual values for PCOS and non-PCOS cases. The classification report provides a comprehensive overview of precision, recall, and F1-score, enabling a deeper understanding of the model's effectiveness in balancing false positives and false negatives.

Key performance metrics such as precision, recall, and F1-score are calculated explicitly. Precision measures the proportion of true positive predictions among all positive predictions, reflecting the model's reliability in identifying PCOS cases accurately. Recall, also known as sensitivity, measures the proportion of actual positives correctly identified, highlighting the model's effectiveness in capturing PCOS cases. The F1-score, which combines precision and recall into a single metric, provides a balanced evaluation of the classifier's performance.

A significant performance highlight is the ROC-AUC (Receiver Operating Characteristic - Area Under Curve), a metric that evaluates the model's ability to discriminate between classes. A higher ROC-AUC score indicates better performance, with the curve plotting the true positive rate (TPR) against the false positive rate (FPR) at various thresholds. The plotted ROC curve showcases the model's excellent discrimination capability, achieving an AUC of 0.950, which indicates near-perfect classification ability.

The Random Forest classifier outperformed Logistic Regression and Support Vector Machines (SVM), with an accuracy of 90.18%. Its F1-score of 0.8367 indicates a superior balance between precision and recall compared to other models. It also achieved the highest precision (0.9111), signifying its high accuracy in correctly predicting PCOS-positive cases. Despite not having the highest recall, it captured a significant portion of actual positives (0.7736), demonstrating overall reliable performance.

The confusion matrix and ROC curve visually reinforce these metrics, showcasing the model's capability in correctly predicting the majority of cases while minimizing misclassifications. The inclusion of all these evaluation techniques, alongside the insights derived from accuracy, precision, recall, F1-score, and AUC-ROC, confirms the Random Forest model as the best performer among the evaluated classifiers. This comprehensive evaluation underscores its effectiveness in handling complex datasets with multiple features, making it an ideal choice for this predictive task.

4.3 SVM

Support Vector Machines (SVM) is a powerful supervised learning algorithm used for classification and regression tasks, designed to find an optimal hyperplane that separates data points of different classes in a high-dimensional space. The goal of SVM is to maximize the margin, which is the distance between the hyperplane and the nearest data points from each class, called support vectors. This approach ensures that the model generalizes well to unseen data. SVM can handle both linear and nonlinear classification problems. For nonlinear problems, it employs kernel functions, such as the Radial Basis Function (RBF), polynomial, or sigmoid kernels, to map input features into higher-dimensional spaces where a linear hyperplane can effectively separate the data.

The provided code demonstrates the implementation of an SVM classifier for predicting whether a patient has PCOS or not. The SVM model is initialized with `probability=True`, enabling it to provide probability estimates for each class, which is particularly useful for evaluating performance metrics like the ROC curve and AUC score. The model is trained on the scaled training data (`X_train_scaled` and `y_train`), ensuring that all features have a uniform scale, as SVM is sensitive to feature magnitudes. This scaling step improves the optimization process and enhances the model's ability to find the optimal hyperplane.

After training, the model is used to predict the class labels (`y_pred`) for the test data (`X_test_scaled`). Additionally, probability estimates for the positive class are extracted using the `predict_proba` method. These probability estimates are essential for calculating the ROC-AUC score, which measures the model's ability to discriminate between the two classes at various thresholds.

The classifier's performance is evaluated using several metrics. Accuracy is calculated as the proportion of correct predictions out of all predictions, providing an overall measure of model performance. Precision is computed to assess the accuracy of positive predictions, reflecting how many of the predicted PCOS cases are true positives. Recall, also known as sensitivity, measures the proportion of actual PCOS cases correctly identified by the model. The F1 score, a harmonic mean of precision and recall, provides a single metric to evaluate the trade-off between these two measures. The classification report summarizes these metrics, providing a detailed breakdown of the model's performance for each class.

A confusion matrix is generated to visualize the classifier's predictions against the actual labels, with the diagonal elements representing correct classifications. This matrix is plotted as a heatmap using Seaborn, with distinct labels for "No PCOS" and "PCOS" classes, making it easier to interpret classification errors. The confusion matrix helps identify specific areas where the model may be underperforming, such as false positives or false negatives.

One of the highlights of the evaluation process is the ROC curve, which plots the true positive rate (sensitivity) against the false positive rate (1-specificity) at various threshold values. The ROC curve provides insights into the model's discrimination capability, with the area under the curve (AUC) serving as a summary metric. A high AUC score, as seen in this case, indicates excellent performance in distinguishing between the two classes. The plotted ROC curve also includes a diagonal line, representing a random classifier's performance, which serves as a baseline for comparison.

Overall, the SVM model showcases its strengths as a versatile and effective classifier capable of handling complex datasets. Its ability to find an optimal hyperplane, combined with the use of probability estimates and comprehensive evaluation metrics, makes it a robust choice for binary classification problems like PCOS prediction. The inclusion of visualizations like the confusion matrix and ROC curve further enhances the interpretability and diagnostic capability of the analysis, providing a holistic view of the model's performance.

4.4 SMOTE ANALYSIS

SMOTE (Synthetic Minority Oversampling Technique) is a widely used approach for addressing the challenge of imbalanced datasets, a common problem in machine learning where one class significantly outnumbers the other. This imbalance can lead to biased models that perform poorly on the minority class, as they tend to predict the majority class more frequently.

SMOTE works by creating synthetic samples of the minority class rather than simply duplicating existing instances. It generates these new samples by interpolating between a minority class sample and one of its nearest neighbors in feature space. This method ensures that the synthetic instances are plausible, lying within the distribution of the minority class. By doing so, SMOTE effectively balances the dataset, allowing machine learning algorithms to better learn the patterns associated with both classes.

It is particularly beneficial for algorithms like logistic regression, decision trees, and neural networks, which are sensitive to class distribution. However, SMOTE is not without limitations. It can sometimes introduce noise, especially if the minority class contains outliers, as these outliers can lead to less realistic synthetic samples. Additionally, SMOTE may not always address the issue of overlapping classes, where the majority and minority classes share similar feature spaces, potentially leading to misclassification. Despite these challenges, SMOTE remains a powerful and accessible technique for improving model performance in classification tasks involving imbalanced data, making it a standard tool in the data scientist's toolkit.

4.4.1 Smote Analysis used in Logistic Regression

In the context of logistic regression, SMOTE (Synthetic Minority Oversampling Technique) is employed to address class imbalance in the dataset, particularly when the target variable (y) has significantly fewer instances of one class compared to the other. For example, in a PCOS detection scenario where the target variable represents PCOS status (1 for PCOS, 0 for No PCOS), an imbalanced dataset might have far fewer instances labeled as 1. This imbalance can result in a logistic regression model biased toward the majority class (0), leading to poor predictive performance for the minority class (1). Metrics such as recall and F1-score for the minority class may be severely impacted due to the lack of sufficient data representing that class.

SMOTE mitigates this issue by generating synthetic samples of the minority class (1) to balance the dataset. These synthetic samples are created by interpolating between existing minority class samples and their nearest neighbors in feature space. By doing so, SMOTE ensures that the minority class is sufficiently represented without merely duplicating existing instances, which could otherwise lead to overfitting.
Once the dataset is balanced using SMOTE, logistic regression is applied to the training data. The balanced data allows the logistic regression algorithm to learn the decision boundary more effectively, capturing patterns and features associated with both classes equally. This results in a model that is not overly biased toward the majority class and can make more accurate predictions for the minority class.

The improvement is particularly evident in metrics like recall, which measures the model's ability to identify true positive instances (correctly predicting PCOS cases). Additionally, the F1-score, which is the harmonic mean of precision and recall, benefits from the balanced data, reflecting better overall performance for the minority class. By enhancing the representation of the minority class through SMOTE, logistic regression becomes more robust in handling imbalanced datasets, ensuring that predictions for both classes are reliable and actionable.

4.4.2 Smote Analysis used in Random Forest Classifier

In the context of a random forest classifier, SMOTE (Synthetic Minority Oversampling Technique) is employed to address class imbalance in the dataset, particularly when the target variable (y) has significantly fewer instances of one class compared to the other. For example, in a PCOS detection scenario where the target variable represents PCOS status (1 for PCOS, 0 for No PCOS), an imbalanced dataset might have far fewer instances labeled as 1. This imbalance can result in a random forest model biased toward the majority class (0), leading to poor predictive performance for the minority class (1). Metrics such as recall and F1-score for the minority class may be severely impacted due to the lack of sufficient data representing that class.

SMOTE mitigates this issue by generating synthetic samples of the minority class (1) to balance the dataset. These synthetic samples are created by interpolating between existing minority class samples and their nearest neighbors in feature space. By doing so, SMOTE ensures that the minority class is sufficiently represented without merely duplicating existing instances, which could otherwise lead to overfitting or fail to provide diversity in the training data.

Once the dataset is balanced using SMOTE, the random forest classifier is trained on this augmented data. By providing a balanced representation of both classes, the random forest algorithm can construct decision trees that give equal weight to each class, leading to improved performance for the minority class. This helps in learning decision boundaries that better separate the two classes and ensures that predictions for the minority class are not overlooked during the ensemble process.

The improvement is particularly evident in metrics like **recall**, which measures the model's ability to identify true positive instances (correctly predicting PCOS cases). Furthermore, the **F1-score**, which is the harmonic mean of precision and recall, benefits significantly from the balanced data, reflecting better overall performance for the minority class. By enhancing the representation of the minority class through SMOTE, the random forest classifier becomes more capable of handling imbalanced datasets, ensuring that predictions are both accurate and equitable for both classes.

4.4.3 Smote Analysis used in Support Vector Machine Classifier

In the context of an SVM (Support Vector Machine) classifier, SMOTE (Synthetic Minority Oversampling Technique) is employed to address class imbalance in the dataset, particularly when the target variable (y) has significantly fewer instances of one class compared to the other. For example, in a PCOS detection scenario where the target variable represents PCOS status (1 for PCOS, 0 for No PCOS), an imbalanced dataset might have far fewer instances labeled as 1. This imbalance can result in an SVM model biased toward the majority class (0), leading to poor predictive performance for the minority class (1). Metrics such as recall and F1-score for the minority class may be severely impacted due to the lack of sufficient data representing that class.

SMOTE mitigates this issue by generating synthetic samples of the minority class (1) to balance the dataset. These synthetic samples are created by interpolating between existing minority class samples and their nearest neighbors in feature space. By doing so, SMOTE ensures that the minority class is sufficiently represented without merely duplicating existing instances, which could otherwise lead to overfitting or fail to capture the diversity of the minority class.

Once the dataset is balanced using SMOTE, the SVM classifier is trained on this augmented data. A balanced dataset allows the SVM to identify a hyperplane that better separates the two classes in the feature space. Since SVMs aim to maximize the margin between classes, the presence of balanced class representations ensures that the minority class contributes equally to the determination of the optimal hyperplane. This reduces the risk of misclassifying minority class samples and results in a more equitable decision boundary.

The improvement is particularly evident in metrics like recall, which measures the model's ability to identify true positive instances (correctly predicting PCOS cases). Additionally, the F1-score, which is the harmonic mean of precision and recall, benefits from the balanced data, reflecting better overall performance for the minority class. By enhancing the representation of the minority class through SMOTE, the SVM classifier becomes more robust in handling imbalanced datasets, ensuring that predictions are both accurate and fair for both classes.

5 RESULTS

5.1 DATASET

The dataset comprises clinical and physical parameters collected from 541 patients across 10 hospitals in Kerala, India, focusing on identifying factors associated with PCOS and infertility. It includes key variables such as `PCOS (Y/N)` diagnosis status, hormonal levels like `beta-HCG` (measured in two phases) and `AMH`, alongside other clinical data. Approximately 32.7% of participants were diagnosed with PCOS, while 67.3% were not. Hormonal data revealed significant variation, with average `beta-HCG` levels of ~664.55 mIU/mL (Phase I) and ~238.23 mIU/mL (Phase II), with some outliers exhibiting values as high as 32,460.97 mIU/mL in Phase I. The `AMH(ng/mL)` levels, indicative of ovarian reserve, ranged from 0.1 ng/mL to 66 ng/mL, with a mean of 5.62 ng/mL, showing diversity in ovarian health among participants. An additional file provided structured instructions for data collection but contained minimal clinical data. Key challenges included cleaning inconsistencies in the `AMH` data and addressing extreme values in hormonal levels. Future analysis should explore correlations between hormonal levels and PCOS diagnosis, investigate regional patterns across hospitals, and address any outliers for a robust understanding of the clinical markers linked to PCOS.

5.2 RESULTS AND DISCUSSION

RESULTS BEFORE SMOTE ANALYSIS

• **Increasing Order of models performance based on Accuracy :-** Logistic Regression(85.89%) -> SVM(87.73%) -> Random Forest Classifier(90.18%).

• **The Random Forest Classifier model is performing the best among the three models based on the following considerations:**

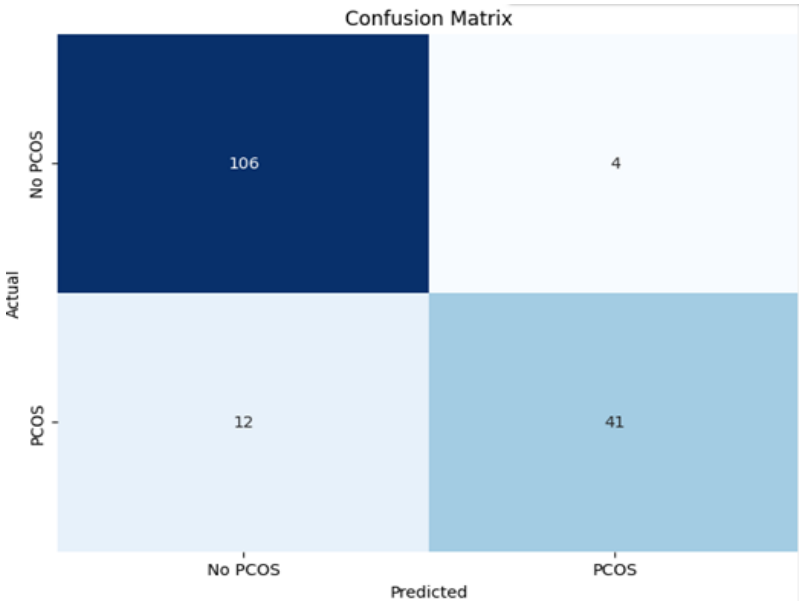
-> **Highest F1 Score:** 0.8367, indicating a good balance between precision and recall.

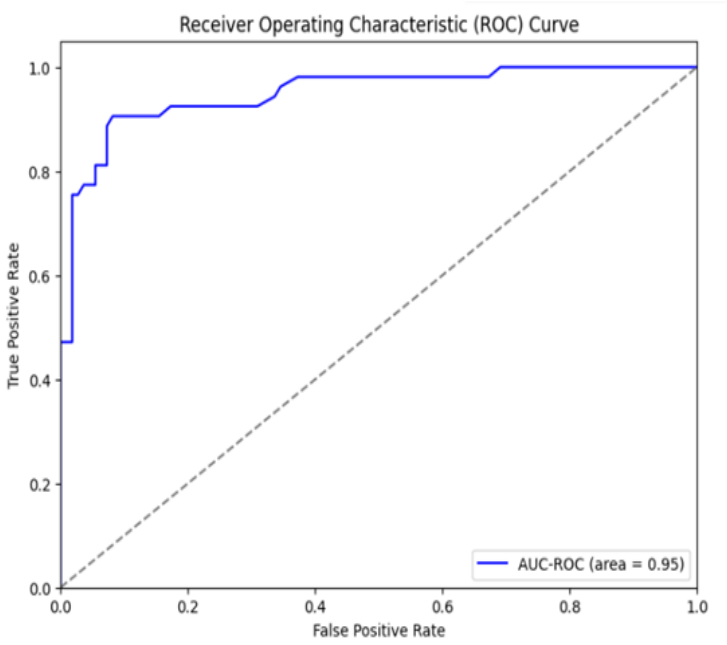
-> **Highest Precision:** 0.9111, showing the highest accuracy in predicting true positives.

-> **High Recall:** 0.7736, which is the second-highest recall score, indicating it captures a significant number of actual positives.

-> **Highest ROC AUC Score:** 0.950, suggesting excellent discrimination capability between classes**.**

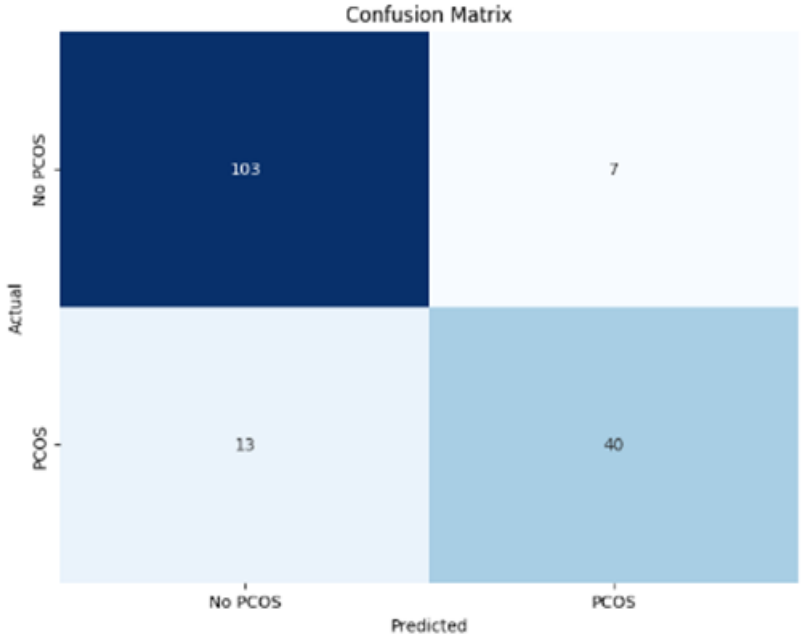
• **The Confusion Matrix and ROC Curve :-**

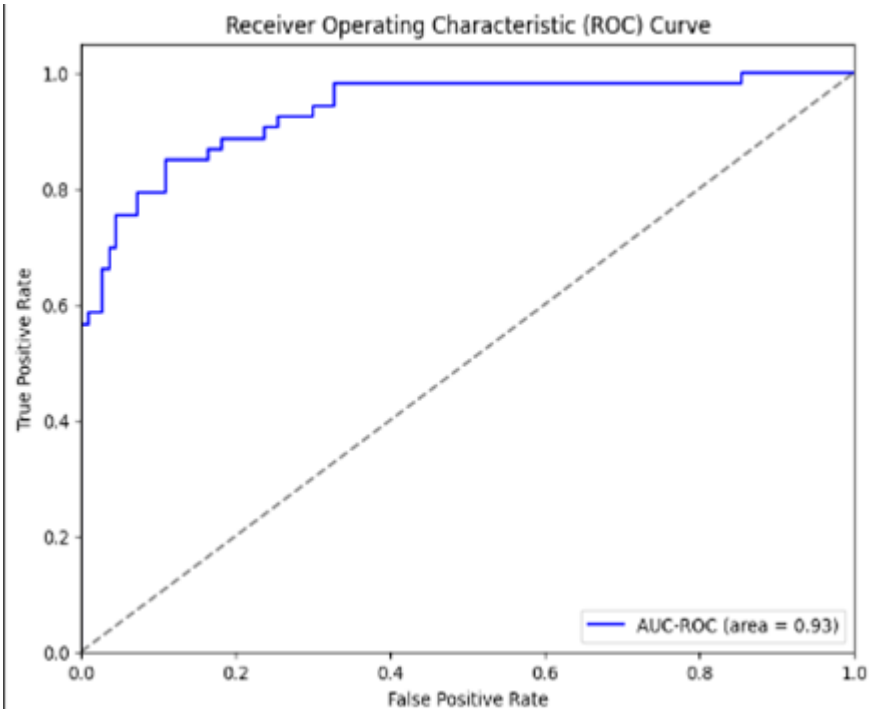




• **SVM comes in the second place in terms of its performance:**
 -> Precision: 0.8511
 -> Recall: 0.7547
 -> F1 Score: 0.8000
 -> Accuracy: 0.8773

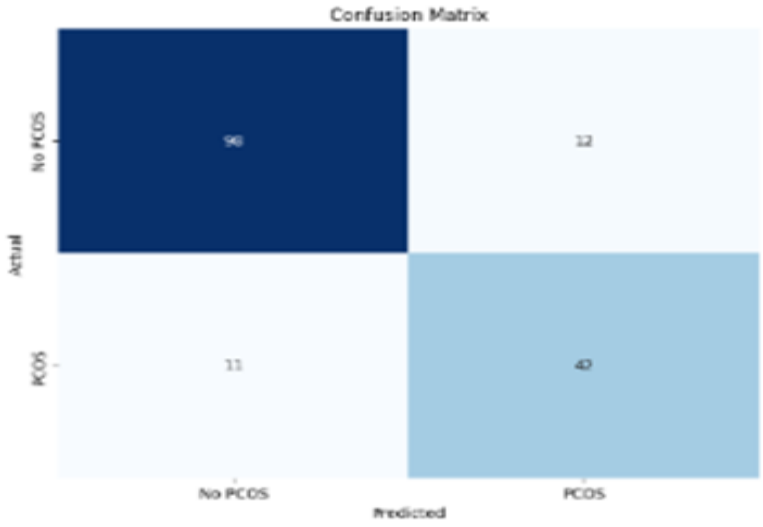
• **The Confusion Matrix and ROC Curve**





• **Logistic Regression comes last terms of its performance:**
 -> Precision: 0.7778
 -> Recall: 0.7925
 -> F1 Score: 0.7850
 -> Accuracy: 0.8589

• **The Confusion Matrix and ROC Curve:**



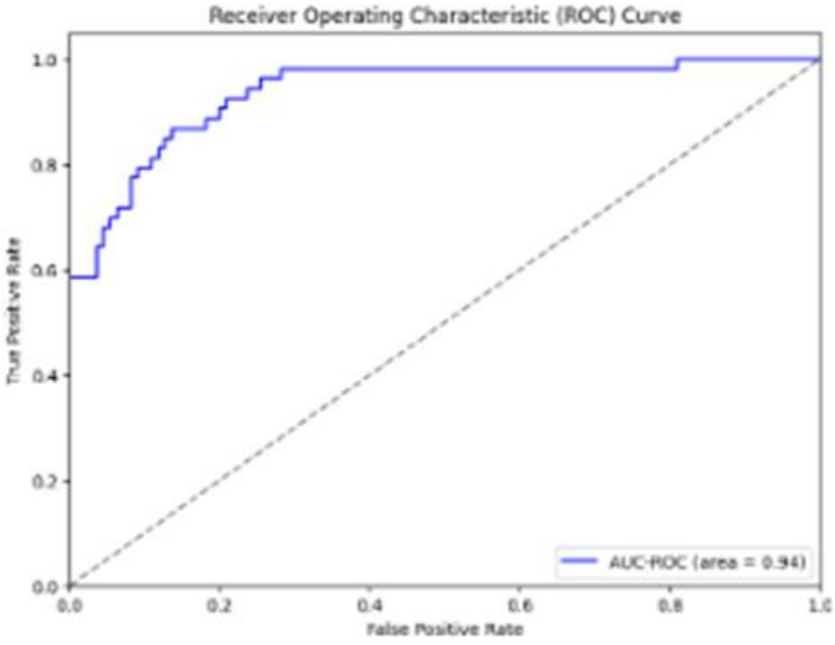
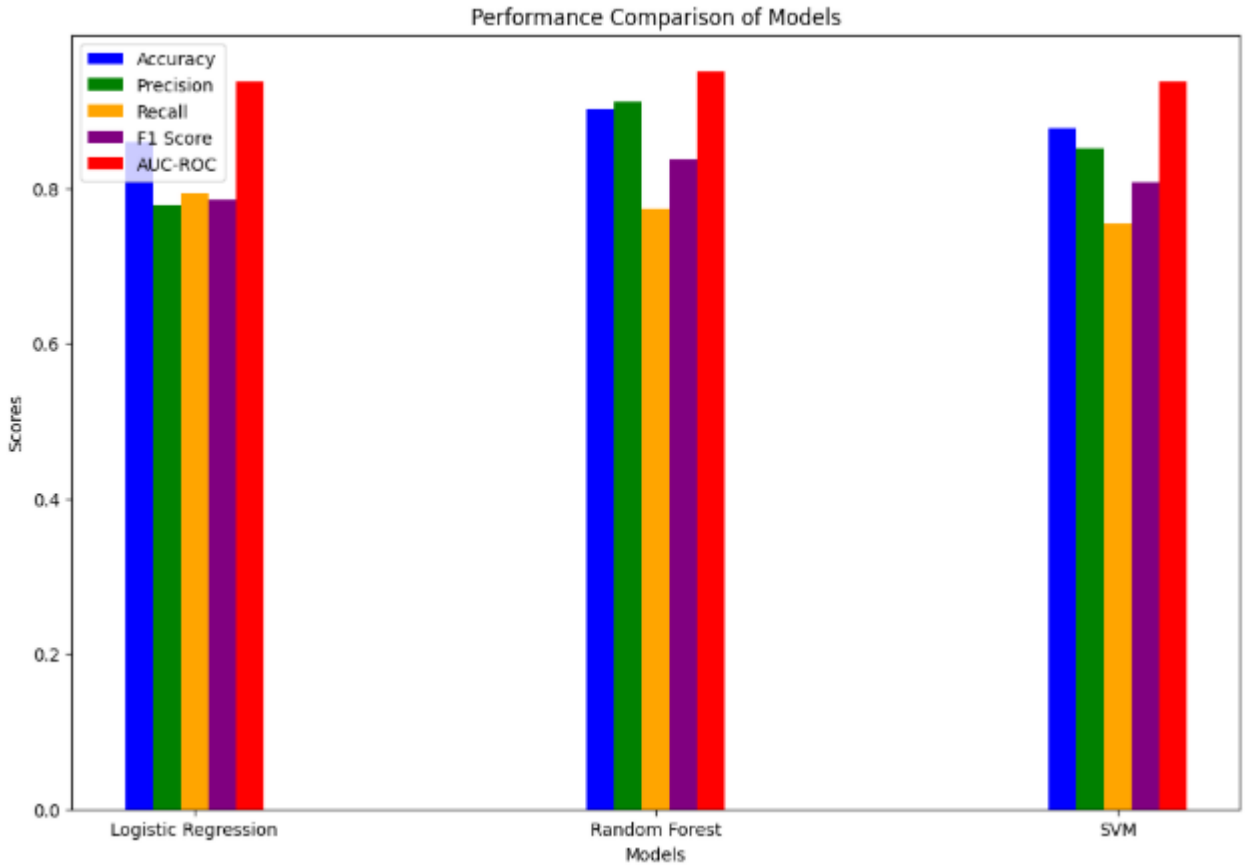


Table 1: Accuracy of Models

Model	Accuracy%
Logistic Regression Accuracy on Test Set	85.89
Random Forest Classifier Accuracy on Test Set	90.18
SVM Classifier Accuracy on Test Set	87.73



Key Insight: Random Forest outperforms in accuracy (0.9018), precision (0.9111), and AUC-ROC (0.9500), highlighting its effectiveness for this dataset.

RESULTS BEFORE SMOTE ANALYSIS

• **Increasing Order of models performance based on Accuracy :-** SVM(85.89%) -> Logistic Regression(87.12%) -> Random Forest Classifier(91.41%).

• The Random Forest Classifier model is performing the best among the three models based on the following considerations:

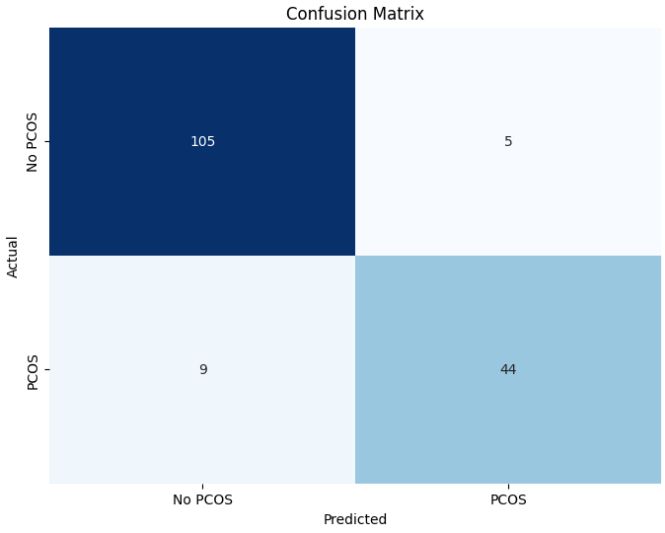
-> **Highest F1 Score:** 0.8627, indicating a good balance between precision and recall.

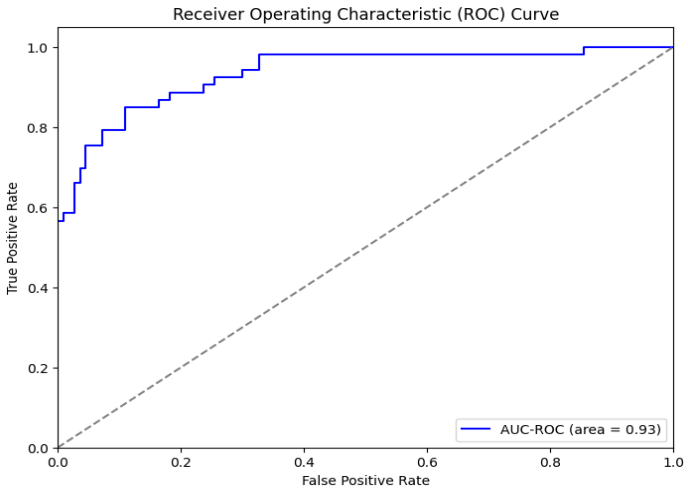
-> **Highest Precision:** 0.8980, showing the highest accuracy in predicting true positives.

-> **High Recall:** 0.8302, which is the second-highest recall score, indicating it captures a significant number of actual positives.

-> **Highest ROC AUC Score:** 0.9444, suggesting excellent discrimination capability between classes**.**

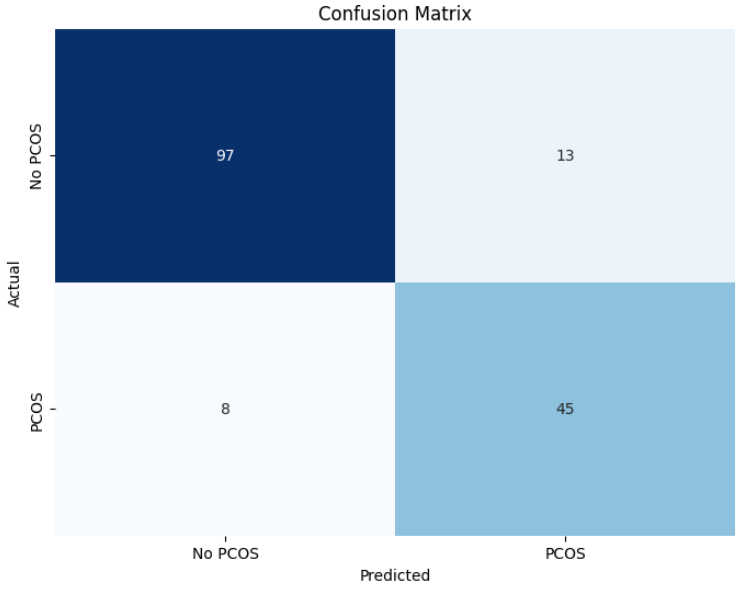
• The Confusion Matrix and ROC Curve :-

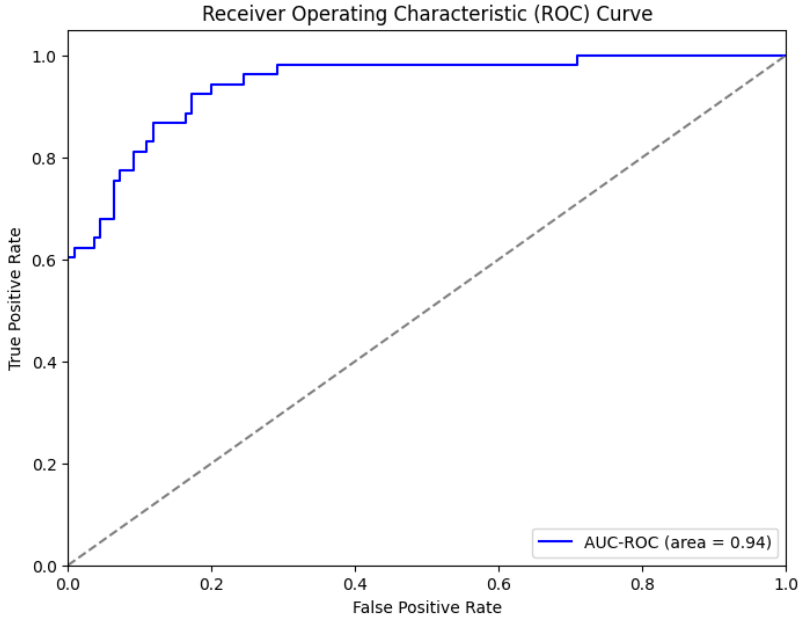




• **Logistic Regression comes in the second place in terms of its performance:**
-> Precision: 0.7759
-> Recall: 0.8491
-> F1 Score: 0.8108
-> Accuracy: 0.8712

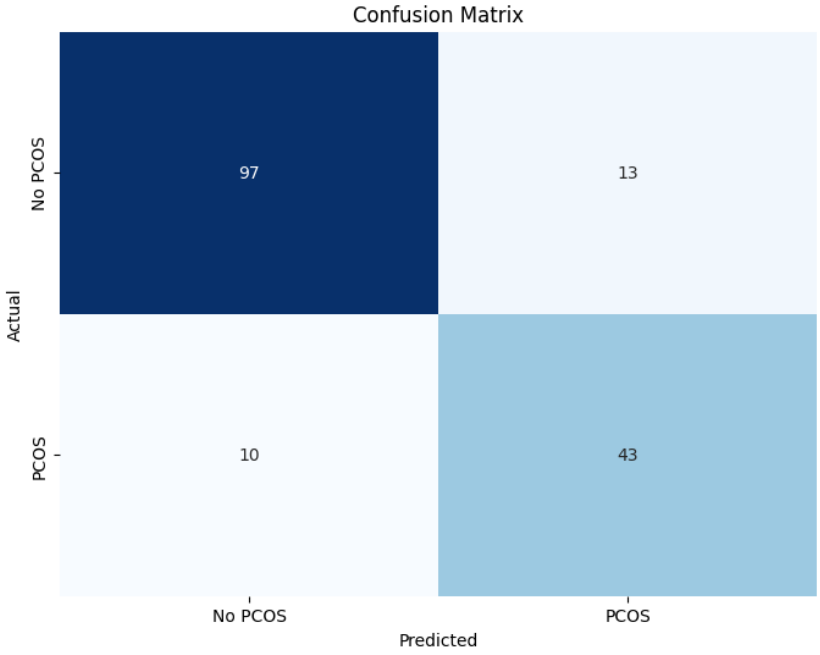
• **The Confusion Matrix and ROC Curve:**





• **SVM comes last in terms of its performance:**
-> Precision: 0.7679
-> Recall: 0.8113
-> F1 Score: 0.7890
-> Accuracy: 0.8589

• **The Confusion Matrix and ROC Curve:**



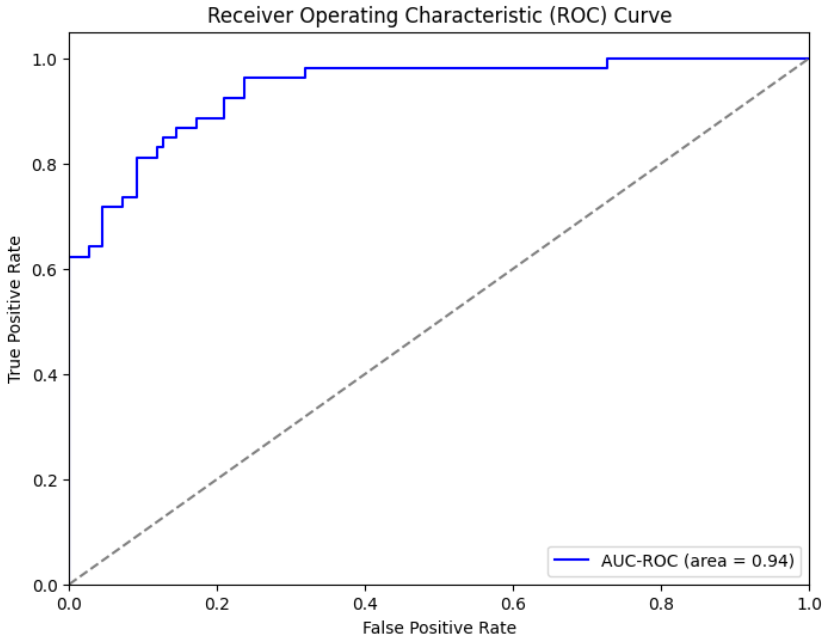
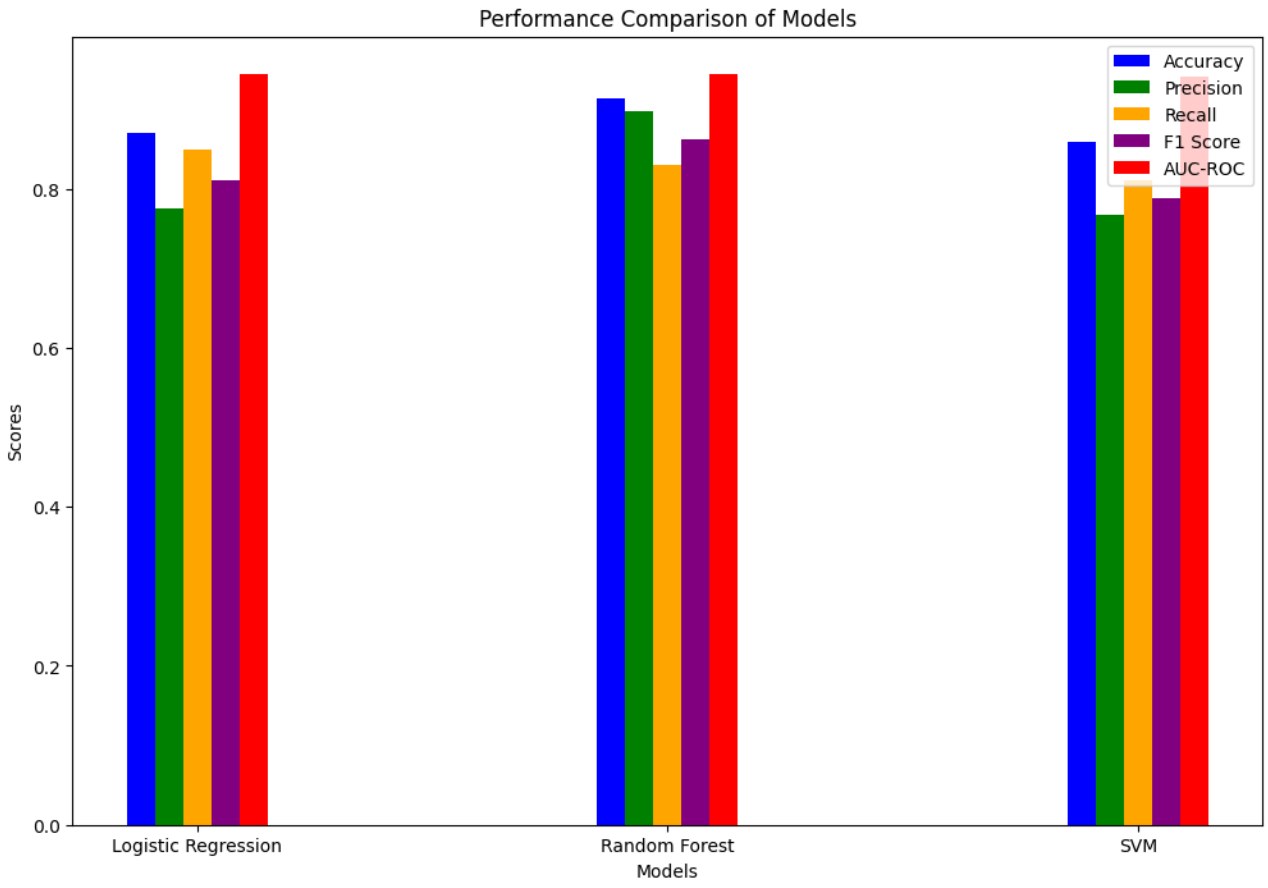


Table 2: Accuracy of Models

Model	Accuracy%
Logistic Regression Accuracy on Test Set	87.12
Random Forest Classifier Accuracy on Test Set	91.41
SVM Classifier Accuracy on Test Set	85.89



Result Analysis After SMOTE

Evaluates the impact of SMOTE (Synthetic Minority Oversampling Technique) on the performance of Logistic Regression, Random Forest, and SVM.

Metrics Analyzed:

Accuracy: Logistic Regression improved slightly (0.8712), Random Forest leads (0.9141).

Precision: Random Forest maintains high precision (0.8980).

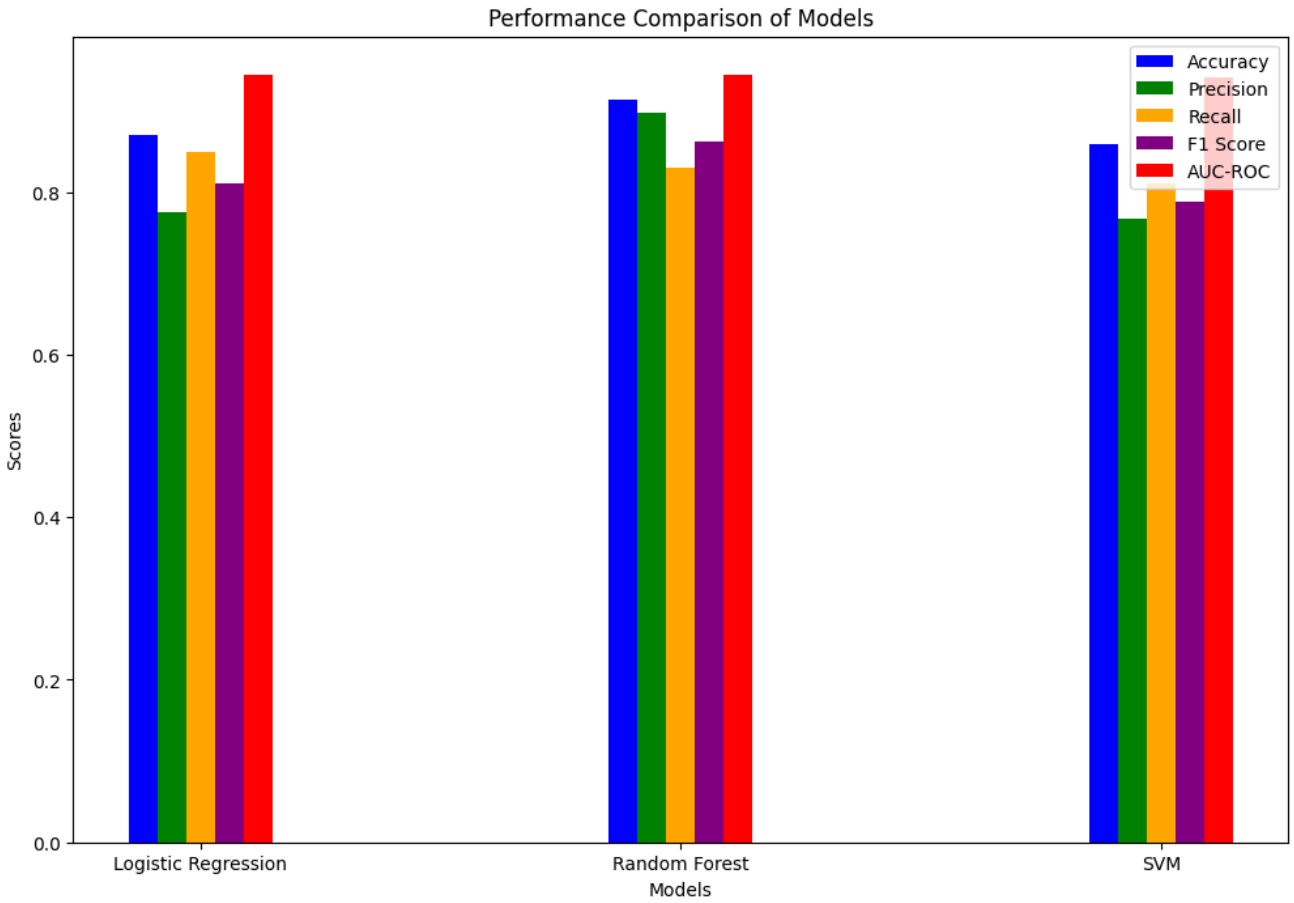
Recall: Logistic Regression excels in recall (0.8491).

F1 Score: Random Forest shows the highest F1 score (0.8627).

AUC-ROC: All models demonstrate competitive AUC-ROC (~0.944).

Visualization: Bar chart allows direct metric comparison, emphasizing the balanced improvement across models post-SMOTE**.**

Key Insight: SMOTE enhances recall and balances performance, with Random Forest remaining the overall best performer in most metrics.



Conclusion

From the Result Analysis we conclude that Random Forest Classifier is the best model to be considered as it is giving the highest Accuracy and Precision than the other models (Logistic Regression, SVM.

Output to User Interactive Test Case

Welcome to the PCOS Prediction System!

Please enter the following details:

Enter value for Sl. No: 3

Enter value for Patient File No.: 3

Enter value for Age (yrs): 33

Enter value for Weight (Kg): 68.8

Enter value for Height(Cm): 165

Enter value for BMI: 25.27089073

Enter value for Blood Group: 11

Enter value for Pulse rate(bpm): 72

Enter value for RR (breaths/min): 18

Enter value for Hb(g/dl): 11.8

Enter value for Cycle(R/I): 2

Enter value for Cycle length(days): 5

Enter value for Marraige Status (Yrs): 10

Enter value for Pregnant(Y/N): 1

Enter value for No. of aborptions: 0

Enter value for I beta-HCG(mIU/mL): 494.08

Enter value for II beta-HCG(mIU/mL): 494.08

Enter value for LH(mIU/mL): 0.88

Enter value for Hip(inch): 40

Enter value for Waist(inch): 36

Enter value for Waist:Hip Ratio: 0.9

Enter value for TSH (mIU/L): 2.54

Enter value for AMH(ng/mL): 6.63

Enter value for PRL(ng/mL): 10.52

Enter value for Vit D3 (ng/mL): 49.7

Enter value for RBS(mg/dl): 84

Enter value for Weight gain(Y/N): 1

```
Enter value for hair growth(Y/N): 1
Enter value for Skin darkening (Y/N): 1
Enter value for Hair loss(Y/N): 1
Enter value for Pimples(Y/N): 1
Enter value for Fast food (Y/N): 1
Enter value for Reg.Exercise(Y/N): 0
Enter value for BP _Systolic (mmHg): 120
Enter value for BP _Diastolic (mmHg): 80
Enter value for Follicle No. (L): 13
Enter value for Follicle No. (R): 15
Enter value for Avg. F size (L) (mm): 18
Enter value for Avg. F size (R) (mm): 20
Enter value for Endometrium (mm): 10
Enter value for tsne-2d-one: 2
Enter value for tsne-2d-two: 2

Prediction: The patient is likely to have PCOS.
Probability of having PCOS: 0.87
Probability of not having PCOS: 0.1
```

6 CONCLUSION AND FUTURE SCOPE

The PCOS detection project utilizes machine learning techniques to identify patterns in patient data, enabling early and accurate diagnosis of Polycystic Ovary Syndrome (PCOS). Through comprehensive data preprocessing, including handling missing values, normalizing features, and eliminating irrelevant columns, the dataset was prepared for effective analysis. Three machine learning models—Logistic Regression, Random Forest Classifier, and Support Vector Machine (SVM)—were employed to classify patients into PCOS-positive and PCOS-negative groups. Among these models, the Random Forest Classifier demonstrated the best performance in terms of accuracy, precision, recall, and F1-score, making it the most reliable model for this dataset. The AUC-ROC curve further validated the performance of each model, with Random Forest consistently showing superior discrimination between the two classes. This project underscores the power of data-driven approaches in healthcare, showcasing how machine learning can assist clinicians in making informed, evidence-based decisions. Future work could explore the integration of more advanced algorithms, such as deep learning or ensemble methods, to further improve prediction accuracy, particularly in addressing imbalanced datasets. Additionally, expanding the dataset to include more diverse features, such as genetic markers, lifestyle factors, or hormonal data, could enhance the model's predictive capabilities. Incorporating explainable AI techniques would also provide greater transparency and interpretability, allowing healthcare providers to better understand the rationale behind model predictions. Such advancements could pave the way for more personalized treatment plans, improved patient outcomes, and broader applicability in PCOS management and other medical conditions.

Key points related to Future Scope of Our Project :

Dataset Expansion: Include diverse features like genetic markers, lifestyle factors, or hormonal data to enhance predictive capabilities.

User-Friendly Interface: Develop a chatbot or web-based front-end for easy interaction with the model, allowing healthcare providers and patients to access predictions seamlessly.

Personalized Treatment Plans: Use AI models to support personalized treatment plans, improving patient outcomes and clinical decisions.

Real-Time Data Analysis: Incorporate real-time data collection from patient devices for continuous monitoring and timely interventions.

7 REFERENCES

1. https://pmc.ncbi.nlm.nih.gov/articles/PMC10137609/

2. https://www.google.com/url?q=https%3A%2F%2Fwww.kaggle.com%2Fdatasets%2Fprasoonkottarathil%2Fpolycystic-ovary-syndrome-pcos

3. https://www.youtube.com/watch?v=DPry9UKAnf4

4. https://chatgpt.com/c/67488463-f9c8-800f-a502-7f12e6d6b9ce

ANNEXURE

1 Importing required modules and libraries

```
!pip install shap
!pip install scikit-optimize
!pip install ydata-profiling
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.manifold import TSNE
from sklearn.preprocessing import StandardScaler
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC
import shap
from sklearn.model_selection import KFold
import numpy as np
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.decomposition import PCA
from sklearn.pipeline import Pipeline
from sklearn.feature_selection import RFE
from ydata_profiling import ProfileReport
```

2 Loading Datasets

```
#Loading the datasets
PCOS_inf = pd.read_csv("/content/PCOS_infertility.csv")
PCOS_woinf = pd.read_excel("/content/PCOS_data_without_infertility.xlsx",sheet_name="Full_new")
```

3 Data Preprocessing

```
#Merging the two files based on patient file no
data = pd.merge(PCOS_woinf,PCOS_inf, on='Patient File No.',suffixes=('','_y'),how='left')
data.head()

#Dropping the repeated features after merging
data =data.drop(['Unnamed: 44', 'Sl. No_y', 'PCOS (Y/N)_y', '  I   beta-HCG(mIU/mL)_y','II       beta-
HCG(mIU/mL)_y', 'AMH(ng/mL)_y'], axis=1)
len(data)
data.head()
```

4 Pandas Profiling

```
profile = ProfileReport(data, title="PCOS Patient Dataset Profiling Report", explorative=True)
profile.to_notebook_iframe()
profile.to_file("PCOS_dataset_profiling_report.html")

data.info()

#data with dtype as object
data["II    beta-HCG(mIU/mL)"].head()
data["AMH(ng/mL)"].head()

#Dealing with categorical values
#Converting the objects into numeric values
data["AMH(ng/mL)"] = pd.to_numeric(data["AMH(ng/mL)"], errors='coerce')
data["II    beta-HCG(mIU/mL)"] = pd.to_numeric(data["II    beta-HCG(mIU/mL)"], errors='coerce')

#Dealing with missing values
columns_with_na = data.columns[data.isna().any()].tolist()
print("Columns containing NA values:")
print(columns_with_na)
```

```
#Filling NA values with the median of that feature
data['Marraige Status (Yrs)'].fillna(data['Marraige Status (Yrs)'].median(),inplace=True)
data['II    beta-HCG(mIU/mL)'].fillna(data['II    beta-HCG(mIU/mL)'].median(),inplace=True)
data['AMH(ng/mL)'].fillna(data['AMH(ng/mL)'].median(),inplace=True)
data['Fast food (Y/N)'].fillna(data['Fast food (Y/N)'].median(),inplace=True)

#Clearing up extra space in the column names
data.columns = [col.strip() for col in data.columns]
```

4 Exploratory Data Analysis (EDA)

```
data.describe()

#Separate the data based on the PCOS class label
pcos_positive = data[data['PCOS (Y/N)'] == 1]
pcos_negative = data[data['PCOS (Y/N)'] == 0]
print(len(pcos_positive))
print(len(pcos_negative))

# Select features for t-SNE
features = data.drop(columns=['Patient File No.', 'PCOS (Y/N)'])

# Normalize the features
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)

# Apply t-SNE
tsne = TSNE(n_components=2, random_state=42)
tsne_results = tsne.fit_transform(features_scaled)

# Add t-SNE results to the data
data['tsne-2d-one'] = tsne_results[:, 0]
data['tsne-2d-two'] = tsne_results[:, 1]

# Plot the t-SNE results
plt.figure(figsize=(16, 10))
plt.scatter(data[data['PCOS (Y/N)'] == 1]['tsne-2d-one'], data[data['PCOS (Y/N)'] == 1]['tsne-2d-two'],
label='PCOS Positive', alpha=0.5)
plt.scatter(data[data['PCOS (Y/N)'] == 0]['tsne-2d-one'], data[data['PCOS (Y/N)'] == 0]['tsne-2d-two'],
label='PCOS Negative', alpha=0.5)
plt.xlabel('t-SNE Component 1')
```

```
plt.ylabel('t-SNE Component 2')
plt.legend()
plt.title('t-SNE plot of PCOS data')
plt.show()

# Compute the correlation matrix
correlation_matrix = data.corr()["PCOS (Y/N)"].sort_values(ascending=False)

# Ensure any date columns (if present) are converted to numerical format
# Assuming there are no date columns for this dataset, otherwise convert them using a similar method

# Calculate correlation with the target variable 'PCOS (Y/N)'
correlation_with_target = data.corr()['PCOS (Y/N)'].sort_values(ascending=False)

# Display the correlations
print(correlation_with_target)

# Plot the correlations
plt.figure(figsize=(12, 8))
correlation_with_target.plot(kind='bar', figsize=(10, 6), color='blue')
plt.title('Correlation of Features with Target Variable (PCOS (Y/N))')
plt.xlabel('Features')
plt.ylabel('Correlation Coefficient')
plt.show()
import seaborn as sns
import matplotlib.pyplot as plt

def features_pairplot(df):
    # Increase the size of each subplot and control the overall figure size
    pairplot = sns.pairplot(df, diag_kind="kde", height=3, aspect=1.2)
    # Adjust the subplots layout for better spacing
    pairplot.fig.subplots_adjust(top=0.95, wspace=0.3, hspace=0.3)
```

```
    # Set overall figure size
    pairplot.fig.set_size_inches(18, 18)
    # Display the plot
    plt.show()
features_pairplot(data)

# Function to plot histograms
def plot_histograms(data, features, pcos_positive, pcos_negative):
    for feature in features:
        plt.figure(figsize=(12, 6))
        sns.histplot(pcos_positive[feature], color='blue', label='PCOS Positive', kde=True, stat="density")
        sns.histplot(pcos_negative[feature], color='red', label='PCOS Negative', kde=True, stat="density")
        plt.title(f'Distribution of {feature} by PCOS status')
        plt.xlabel(feature)
        plt.ylabel('Density')
        plt.legend()
        plt.show()

# Select a subset of features for initial visualization
features_to_plot = ['Follicle No. (R)','Follicle No. (L)','Skin darkening (Y/N)', 'hair growth(Y/N)','Weight gain(Y/N)','PRG(ng/mL)','FSH(mIU/mL)','FSH/LH']
plot_histograms(data, features_to_plot, pcos_positive, pcos_negative)

# Drop the specified columns
columns_to_drop = ['FSH(mIU/mL)','PRG(ng/mL)','FSH/LH']
data = data.drop(columns=columns_to_drop)

X = data.drop(columns=['PCOS (Y/N)'])
y = data['PCOS (Y/N)']
```

5 Application of Models

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, roc_curve, roc_auc_score
from sklearn.metrics import precision_score, recall_score, f1_score

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42, stratify=y)

# Normalize the features using StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Initialize the Logistic Regression model
log_reg = LogisticRegression()

# Train the model on the training data
log_reg.fit(X_train_scaled, y_train)

# Predict on the test data
y_pred = log_reg.predict(X_test_scaled)
y_pred_proba = log_reg.predict_proba(X_test_scaled)[:, 1]  # Probability estimates for ROC curve

# Evaluate the model performance
accuracy = accuracy_score(y_test, y_pred)
# Calculate precision, recall, and F1-score
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

# Print the precision, recall, and F1-score
print(f'Precision: {precision:.4f}')
print(f'Recall: {recall:.4f}')
```

```
print(f'F1 Score: {f1:.4f}')
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

# Output the results
print(f'Accuracy: {accuracy:.4f}')
print('Classification Report:')
print(class_report)

# Plot the confusion matrix as a heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", cbar=False, xticklabels=["No PCOS",
"PCOS"], yticklabels=["No PCOS", "PCOS"])
plt.title("Confusion Matrix")
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()

# Calculate AUC-ROC
roc_auc = roc_auc_score(y_test, y_pred_proba)

# Plot the AUC-ROC curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', label=f'AUC-ROC (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='grey', linestyle='--')  # Diagonal line
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```

```
# Print AUC-ROC score
print(f'AUC-ROC Score: {roc_auc:.4f}')
```

Logistic Regression with SMOTE-Analysis

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42, stratify=y)

# Normalize the features using StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Apply SMOTE to the training data
smote = SMOTE(random_state=42)
X_train_smote, y_train_smote = smote.fit_resample(X_train_scaled, y_train)

# Check the distribution of the target variable after SMOTE
print("Before SMOTE:", y_train.value_counts())
print("After SMOTE:", pd.Series(y_train_smote).value_counts())

# Initialize the Logistic Regression model
log_reg = LogisticRegression()

# Train the model on the SMOTE-enhanced training data
log_reg.fit(X_train_smote, y_train_smote)

# Predict on the test data (no changes here)
y_pred = log_reg.predict(X_test_scaled)
y_pred_proba = log_reg.predict_proba(X_test_scaled)[:, 1]

# Evaluate the model performance
accuracy = accuracy_score(y_test, y_pred)
# Calculate precision, recall, and F1-score
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
```

```
f1 = f1_score(y_test, y_pred)

# Print the precision, recall, and F1-score
print(f'Precision: {precision:.4f}')
print(f'Recall: {recall:.4f}')
print(f'F1 Score: {f1:.4f}')
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

# Output the results
print(f'Accuracy: {accuracy:.4f}')
print('Classification Report:')
print(class_report)
# Plot the confusion matrix as a heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", cbar=False, xticklabels=["No PCOS",
"PCOS"], yticklabels=["No PCOS", "PCOS"])
plt.title("Confusion Matrix")
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()

# Calculate AUC-ROC
roc_auc = roc_auc_score(y_test, y_pred_proba)

# Plot the AUC-ROC curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', label=f'AUC-ROC (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='grey', linestyle='--')  # Diagonal line
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
```

```
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()

# Print AUC-ROC score
print(f'AUC-ROC Score: {roc_auc:.4f}')
```

Random Forest Classifier

```
# Train a Random Forest classifier
clf = RandomForestClassifier(random_state=42)
clf.fit(X_train_scaled, y_train)

# Predict on the test data
y_pred = clf.predict(X_test_scaled)
y_pred_proba = clf.predict_proba(X_test_scaled)[:, 1]  # Probability estimates for ROC curve

# Evaluate the model performance
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

# Output the results
print(f'Accuracy: {accuracy:.4f}')
# Calculate precision, recall, and F1-score
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

# Print the precision, recall, and F1-score
print(f'Precision: {precision:.4f}')
```

```
print(f'Recall: {recall:.4f}')
print(f'F1 Score: {f1:.4f}')
print('Classification Report:')
print(class_report)
# Plot the confusion matrix as a heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", cbar=False, xticklabels=["No PCOS", "PCOS"], yticklabels=["No PCOS", "PCOS"])
plt.title("Confusion Matrix")
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()

# Calculate AUC-ROC
roc_auc = roc_auc_score(y_test, y_pred_proba)

# Plot the AUC-ROC curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', label=f'AUC-ROC (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='grey', linestyle='--')  # Diagonal line
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()

# Print AUC-ROC score
```

```
print(f'AUC-ROC Score: {roc_auc:.4f}')
```

Random Forest Classifier using Smote-Analysis

```
# Initialize and train the Random Forest model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train_smote, y_train_smote)

# Predict on the test data
y_pred = rf_model.predict(X_test_scaled)

# Evaluate the model performance
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

# Output the results
print(f'Accuracy: {accuracy:.4f}')
# Calculate precision, recall, and F1-score
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

# Print the precision, recall, and F1-score
print(f'Precision: {precision:.4f}')
print(f'Recall: {recall:.4f}')
print(f'F1 Score: {f1:.4f}')
print('Classification Report:')
print(class_report)
# Plot the confusion matrix as a heatmap
plt.figure(figsize=(8, 6))
```

```
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", cbar=False, xticklabels=["No PCOS", "PCOS"], yticklabels=["No PCOS", "PCOS"])
plt.title("Confusion Matrix")
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()

# Calculate AUC-ROC
roc_auc = roc_auc_score(y_test, y_pred_proba)

# Plot the AUC-ROC curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', label=f'AUC-ROC (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='grey', linestyle='--')  # Diagonal line
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()

# Print AUC-ROC score
print(f'AUC-ROC Score: {roc_auc:.4f}')
```

Support Vector Machine Classifier(SVM)

```
from sklearn.svm import SVC
# Initialize the classifier with probability=True
clf_svm = SVC(probability=True, random_state=42)
# Train the model (assuming X_train_scaled and y_train are already defined)
clf_svm.fit(X_train_scaled, y_train)
# Predict on the test data
y_pred = clf_svm.predict(X_test_scaled)
y_pred_proba = clf_svm.predict_proba(X_test_scaled)[:, 1]  # Probability estimates for ROC curve

# Evaluate the model performance
accuracy = accuracy_score(y_test, y_pred)
# Calculate precision, recall, and F1-score
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

# Print the precision, recall, and F1-score
print(f'Precision: {precision:.4f}')
print(f'Recall: {recall:.4f}')
print(f'F1 Score: {f1:.4f}')
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

# Output the results
print(f'Accuracy: {accuracy:.4f}')
print('Classification Report:')
print(class_report)
# Plot the confusion matrix as a heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", cbar=False, xticklabels=["No
```

```
PCOS", "PCOS"], yticklabels=["No PCOS", "PCOS"])
plt.title("Confusion Matrix")
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()

# Calculate AUC-ROC
roc_auc = roc_auc_score(y_test, y_pred_proba)

# Plot the AUC-ROC curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', label=f'AUC-ROC (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='grey', linestyle='--')  # Diagonal line
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()

# Print AUC-ROC score
print(f'AUC-ROC Score: {roc_auc:.4f}')
```

Support Vector Machine Classifier using Smote-Analysis

```
# Initialize and train the SVM model
svm_model = SVC(kernel='linear', random_state=42, probability=True)  # Enable probability=True for predict_proba
svm_model.fit(X_train_smote, y_train_smote)
```

```
# Predict on the test data
y_pred = svm_model.predict(X_test_scaled)
y_pred_proba = svm_model.predict_proba(X_test_scaled)[:, 1]  # Probability estimates for ROC curve

# Evaluate the model performance
accuracy = accuracy_score(y_test, y_pred)
# Calculate precision, recall, and F1-score
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

# Print the precision, recall, and F1-score
print(f'Precision: {precision:.4f}')
print(f'Recall: {recall:.4f}')
print(f'F1 Score: {f1:.4f}')
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

# Output the results
print(f'Accuracy: {accuracy:.4f}')
print('Classification Report:')
print(class_report)
# Plot the confusion matrix as a heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", cbar=False, xticklabels=["No PCOS", "PCOS"], yticklabels=["No PCOS", "PCOS"])
plt.title("Confusion Matrix")
plt.ylabel('Actual')
plt.xlabel('Predicted')
```

```
plt.show()

# Calculate AUC-ROC
roc_auc = roc_auc_score(y_test, y_pred_proba)

# Plot the AUC-ROC curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', label=f'AUC-ROC (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='grey', linestyle='--')  # Diagonal line
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()

# Print AUC-ROC score
print(f'AUC-ROC Score: {roc_auc:.4f}')
```

6 Result Analysis

```
# Define models and their corresponding performance metrics
models = ['Logistic Regression', 'Random Forest', 'SVM']
accuracy_scores = [0.8589, 0.9018, 0.8773]
precision_scores = [0.7778, 0.9111, 0.8511]
recall_scores = [0.7925, 0.7736, 0.7547]
f1_scores = [0.7850, 0.8367, 0.8080]
aucroc_scores = [0.9374, 0.9500, 0.9374]
```

```
# Define the width of each bar and the positions on the x-axis
bar_width = 0.15
x = np.arange(len(models))*2.5

# Create a figure and axis
fig, ax = plt.subplots(figsize=(12, 8))

# Plot bars for each metric
bars1 = ax.bar(x - 2*bar_width, accuracy_scores, bar_width, label='Accuracy', color='b')
bars2 = ax.bar(x - bar_width, precision_scores, bar_width, label='Precision', color='g')
bars3 = ax.bar(x, recall_scores, bar_width, label='Recall', color='orange')
bars4 = ax.bar(x + bar_width, f1_scores, bar_width, label='F1 Score', color='purple')
bars5 = ax.bar(x + 2*bar_width, aucroc_scores, bar_width, label='AUC-ROC', color='red')

# Set x-axis labels and ticks
ax.set_xticks(x)
ax.set_xticklabels(models)

# Set axis labels and title
ax.set_xlabel('Models')
ax.set_ylabel('Scores')
ax.set_title('Performance Comparison of Models')

# Add legend
ax.legend()

# Display the plot
plt.show()
```

Result Analysis after using SMOTE

```
# Define models and their corresponding performance metrics
models = ['Logistic Regression', 'Random Forest', 'SVM']
accuracy_scores = [0.8712, 0.9141, 0.8589]
precision_scores = [0.7759, 0.8980, 0.7679]
recall_scores = [0.8491, 0.8302, 0.8113]
f1_scores = [0.8108, 0.8627, 0.7890]
aucroc_scores = [0.9444, 0.9444, 0.9407]

# Define the width of each bar and the positions on the x-axis
bar_width = 0.15
x = np.arange(len(models))*2.5

# Create a figure and axis
fig, ax = plt.subplots(figsize=(12, 8))

# Plot bars for each metric
bars1 = ax.bar(x - 2*bar_width, accuracy_scores, bar_width, label='Accuracy', color='b')
bars2 = ax.bar(x - bar_width, precision_scores, bar_width, label='Precision', color='g')
bars3 = ax.bar(x, recall_scores, bar_width, label='Recall', color='orange')
bars4 = ax.bar(x + bar_width, f1_scores, bar_width, label='F1 Score', color='purple')
bars5 = ax.bar(x + 2*bar_width, aucroc_scores, bar_width, label='AUC-ROC', color='red')

# Set x-axis labels and ticks
ax.set_xticks(x)
ax.set_xticklabels(models)

# Set axis labels and title
ax.set_xlabel('Models')
ax.set_ylabel('Scores')
ax.set_title('Performance Comparison of Models')
```

```
# Add legend
ax.legend()

# Display the plot
plt.show()
```

User Interactive Test Case

```
# Function for interactive user input and prediction
def predict_pcos():
    print("Welcome to the PCOS Prediction System!")
    print("Please enter the following details:")

    # Collect user inputs
    user_data = {}
    for feature in X.columns:
        while True:
            try:
                value = input(f"Enter value for {feature}: ")
                if data[feature].dtype == 'object':
                    value = pd.to_numeric(value, errors='coerce')
                    if pd.isna(value):
                        raise ValueError("Invalid input. Please enter a numeric value.")
                else:
                    value = float(value)
                user_data[feature] = value
                break
            except ValueError as e:
                print(e)
    # Convert user input into a DataFrame
```

```
    user_df = pd.DataFrame([user_data])

    # Normalize the user input using the same scaler
    user_scaled = scaler.transform(user_df)

    # Make prediction
    prediction = rf_model.predict(user_scaled)[0]
    prediction_proba = rf_model.predict_proba(user_scaled)[0]

    # Display result
    if prediction == 1:
        print("\nPrediction: The patient is likely to have PCOS.")
    else:
        print("\nPrediction: The patient is unlikely to have PCOS.")
    print(f"Probability of having PCOS: {prediction_proba[1]:.2f}")
    print(f"Probability of not having PCOS: {prediction_proba[0]:.2f}")

# Run the interactive prediction
if __name__ == "__main__":
    predict_pcos()
```