Deploying PCOS Machine Learning Model on Cloud

PROJECT REPORT

Submitted in fulfilment for the J Component of **ITA6009 – Cloud Computing**

in

M.C.A

by

Debpritam Roy (22MCA0027)

Kamran Ansari (22MCA0223)

Bhooshan Birje (22MCA0233)

Under the guidance of

Dr. KRISHNAMOORTHY N

SITE

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Information Technology and Engineering

TABLE OF CONTENTS

Abstract

Polycystic Ovary Syndrome (PCOS) is a hormonal imbalance brought on by the ovaries producing too many male hormones, which affects a woman's reproductive system during her childbearing age. PCOS has been identified as of the major factors of infertility in women. It also raises the likelihood of developing other illnesses including diabetes, high blood pressure, sleep difficulties, depression, and anxiety, among others. After experimenting with various classification models, we chose Support Vector Machine (SVM), which gave the best results with 91% accuracy and 81% sensitivity among all the classification models compared. To make this prediction model accessible and available to all, we intend deploy the model on the cloud in this project. But deploying on big cloud providers like AWS and Azure requires understanding large ecosystems and complex services. Most of the cloud apps created, normally require only a minuscule number of services in comparison to the magnitude of services available from these big providers. This gap of expertise is fulfilled by Platform as a Services (PaaS) such as Render. PaaS simplifies the task of deploying cloud apps. It abstracts away the complexities of big cloud providers, providing faster and easier deployment models suitable for most cloud applications. PaaS also provides scalability, flexibility, and cost savings. With PaaS, developers can focus on developing and deploying applications without having to worry about infrastructure management, maintenance, and scaling. PaaS also allows for faster time-to-market and reduces the need for specialized technical knowledge. In this project, we deploy a FastAPI backend API endpoint of the trained model on Render allowing users of frontend client query and get the predictions. On the frontend we have a NextJS app running on another PaaS platform called Vercel. Vercel is a frontend hosting service which provides CI/CD, caching and other hosting optimizations.

Keywords: Polycystic Ovary Syndrome, Support Vector Machine, Platform As A Service, Render, NextJS, Vercel

Introduction

In recent years, the field of machine learning has witnessed remarkable advancements in healthcare, offering innovative solutions to improve diagnostics and treatment outcomes. Polycystic Ovary Syndrome (PCOS) is a prevalent endocrine disorder affecting millions of women worldwide, characterized by hormonal imbalances and ovarian dysfunction. Early detection and timely intervention are crucial for effective management of PCOS and prevention of associated health complications.

This project aims to deploy a PCOS detection machine learning model on the cloud, harnessing the power of scalable computing resources and ensuring accessibility and convenience for healthcare providers and patients alike. By leveraging the capabilities of cloud computing, we seek to revolutionize PCOS diagnostics by delivering a robust and accurate solution that can be readily deployed in diverse clinical settings.

Our machine learning model has been trained on a comprehensive dataset comprising a wide range of patient demographics, clinical attributes, and diagnostic results. By analyzing this data, the model has learned to identify patterns and indicators of PCOS, enabling reliable predictions with high precision. By deploying this model on the cloud, we enable healthcare practitioners to access the diagnostic capabilities remotely, streamlining the diagnostic process and reducing the time required for diagnosis.

Cloud deployment offers numerous advantages for our PCOS detection model. Firstly, it eliminates the need for on-premises infrastructure, reducing costs associated with hardware provisioning, maintenance, and upgrades. Additionally, cloud-based deployment ensures scalability, allowing the model to handle a large number of concurrent requests and accommodate increasing demand as awareness about PCOS grows. Furthermore, the cloud platform offers robust security measures, protecting patient data and ensuring compliance with privacy regulations.

By deploying our PCOS detection model on the cloud, we envision a future where healthcare providers can rapidly and accurately diagnose PCOS, empowering them to develop tailored treatment plans and mitigate the long-term health risks associated with the disorder. Patients will benefit from faster and more accessible diagnostics, enabling early intervention and personalized care.

In the following sections, we will delve into the technical details of our cloud-based PCOS detection system, discussing the architecture, data flow, and deployment strategy. We will also explore the potential impact of this project on PCOS diagnostics and its broader implications for leveraging machine learning in healthcare. Together, we aim to revolutionize the way PCOS is

diagnosed and managed, improving the quality of care and enhancing the overall well-being of affected individuals.

Problem Statement

Polycystic Ovary Syndrome (PCOS) is a prevalent endocrine disorder affecting a significant number of women worldwide. It is characterized by hormonal imbalances, ovarian dysfunction, and various symptoms such as irregular menstrual cycles, excessive hair growth, acne, and weight gain. PCOS not only affects reproductive health but also poses long-term risks such as infertility, type 2 diabetes, cardiovascular diseases, and mental health issues.

The current diagnostic process for PCOS relies on a combination of medical history evaluation, physical examinations, and laboratory tests, including hormone level assessments and ultrasound imaging. However, this process can be time-consuming, subjective, and dependent on the expertise of the healthcare provider. Furthermore, access to specialized PCOS clinics and diagnostic facilities may be limited, especially in remote areas or underserved communities.

There is a pressing need for an accurate, efficient, and scalable diagnostic solution that can enable early detection of PCOS and facilitate timely intervention. Leveraging the power of machine learning and cloud computing, this project aims to address this challenge by developing and deploying a PCOS detection model on the cloud. The primary objective is to create a robust and accessible solution that can assist healthcare providers in diagnosing PCOS with high precision and speed, ultimately improving patient outcomes.

The key problems to be addressed in this project include:

1. **Limited accessibility to PCOS diagnostic facilities**: Access to specialized PCOS diagnostic facilities may be limited in certain regions, leading to delayed diagnoses and potential health complications. A cloud-based PCOS detection model can bridge this gap by providing a remote and accessible solution that can be accessed by healthcare providers regardless of their location.
2. **Time-consuming diagnostic process**: The current diagnostic process for PCOS can be time-consuming, involving multiple appointments and waiting periods for test results. By deploying a machine learning model on the cloud, we aim to reduce the time required for diagnosis, enabling faster intervention and timely management of PCOS.
3. **Infrastructure Provisioning and Management**: Deploying a PCOS detection model on big cloud providers such as AWS and GCP requires careful infrastructure provisioning and management. It involves setting up virtual machines, configuring networking, and

ensuring scalability and availability. This can be a complex task that requires expertise in cloud infrastructure.

4. **Cost Management**: Big cloud providers offer a wide range of services, each with its own pricing model. Deploying and running a machine learning model on these platforms can be cost-intensive, especially when handling large datasets or experiencing high traffic volumes. Optimizing costs and choosing the right resources can be challenging to ensure efficient utilization of cloud resources.
5. **Scalability and Performance**: Ensuring that the PCOS detection model can handle a large number of concurrent requests and provide real-time responses is crucial. Scaling the infrastructure dynamically to meet varying demand while maintaining optimal performance can be challenging. Proper load balancing, auto-scaling configurations, and efficient resource allocation are essential for a successful deployment
6. **Cost-Effectiveness**: SaaS platforms typically offer pay-as-you-go pricing models, enabling cost optimization by charging only for the resources consumed. This can be particularly beneficial for smaller projects or when the demand for the PCOS detection model is variable. It eliminates the need for upfront infrastructure investments and provides cost transparency.
7. **Scalability and Performance Optimization**: SaaS platforms often come with built-in scalability features, automatically handling traffic spikes and scaling resources as needed. This eliminates the need for manual scaling configurations and ensures optimal performance. These platforms often utilize serverless computing, allowing for efficient resource allocation and cost-effective scaling.

By addressing these problems, this project seeks to revolutionize PCOS diagnostics, providing healthcare providers with an accessible, accurate, and efficient tool for early detection and intervention. Through the deployment of a cloud-based machine learning model, we aim to improve the quality of care for individuals with PCOS, mitigating the long-term health risks associated with the disorder and empowering women to take control of their reproductive health.

And by leveraging SaaS platforms like Vercel and Render, deploying the PCOS detection model becomes more streamlined, cost-effective, and developer-friendly. These platforms offer advantages such as simplified deployment, scalability, cost optimization, managed security, and compliance, making them attractive options for deploying machine learning models in a cloud environment.

Modules Description

The modules and methods used in this project are as follows

- **Prediction Model Building**

  Is the phase where the prediction model is built starting from the defining the problem statement to tuning and selecting the best performing model. Various methods used in this step are explained above in architecture diagram.

- **Model Exporting**

  After successfully building and testing the built model, it is ready for deployment. Now we export the model to deploy it in the cloud. For this we use the pickle function from the Python's standard library. We take the trained model and pickle it (save in binary format). Also we pickle the Standard Scaler object which is used to normalise the test values.

- **Building REST API**

  In this phase we build the REST API to server the model predictions. Now we use the saved binary model and unpickle (unpack) to generate predictions. We are building a REST API here using the FastAPI library in Python. We expose the "predict" endpoint which accepts the "POST" requests and responds with the prediction after generating the prediction.

- **Building Frontend**

  In this phase a simple and user friendly frontend is made to facilitate the end users in using the API. It acts as a interface between the model and user. It takes parameters from the user and displays back the prediction after calling the API with that data in the background and receiving the prediction. The website will be written in HTML, CSS and Javascript.

- **Deploying API and Frontend**

  Now we deploy the built API and Frontend website. For this we are using a PaaS platform called Render. It provides easy way to deploy applications with built in CICD capabilities. We deploy API using the Python environment in Render. And serve the Frontend website using static host.

System Design

Prediction Model Architecture

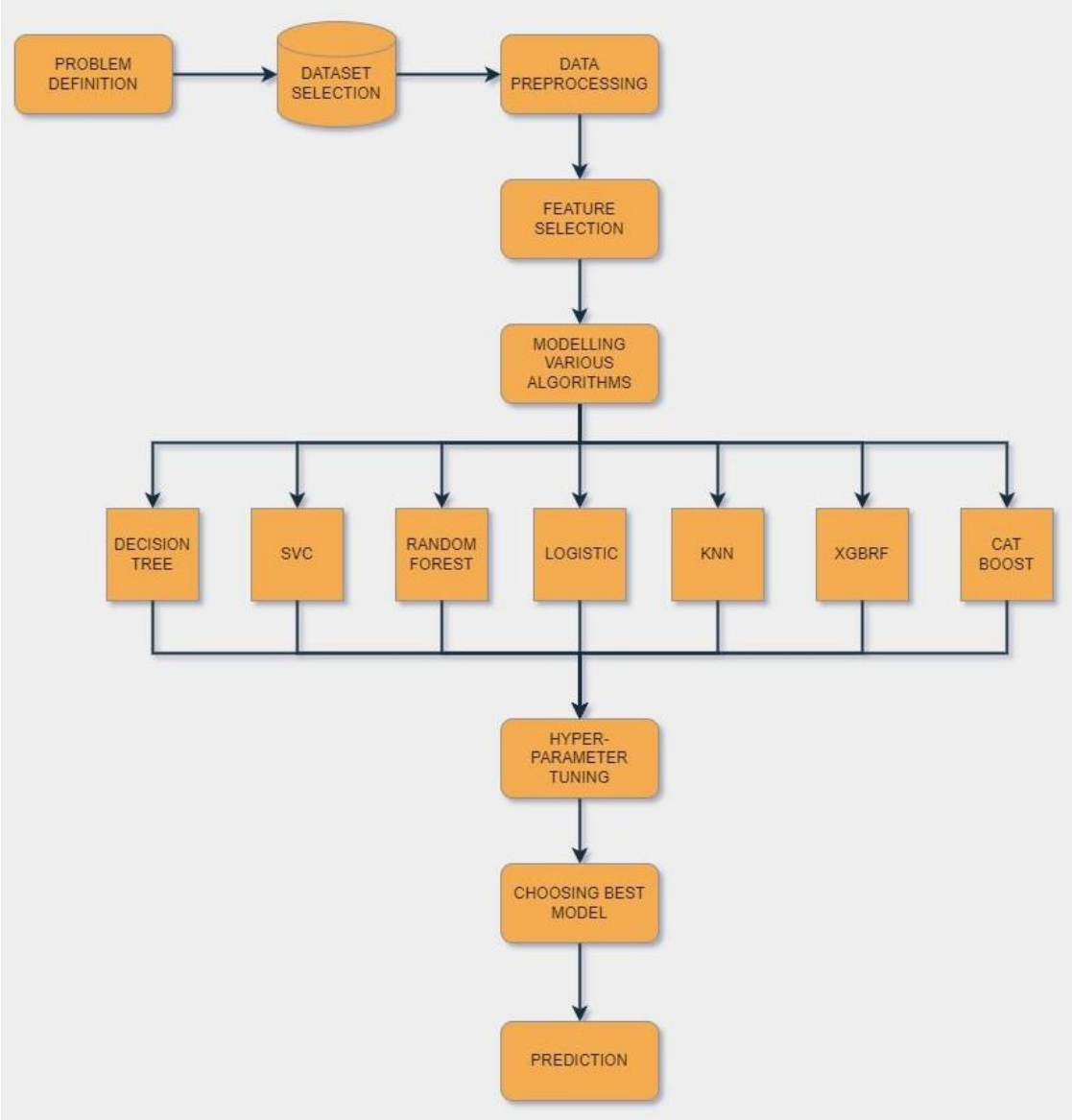


Figure 1 Prediction Model Architecture/Pipeline

Problem Definition: The first stage of the ML pipeline is problem definition, which involves defining the objective of the model and identifying the problem that the model is intended to solve. This stage is crucial because it helps to determine the appropriate ML algorithm to use and the type of data needed to train the model.

Dataset Selection: The second stage of the ML pipeline is dataset selection, which involves selecting the appropriate dataset to train the model. The dataset should contain relevant and

representative data that is sufficient to build the ML model. Dataset selection is an important step because the quality of the data used to train the model significantly affects the accuracy and reliability of the predictions made by the model.

Data Preprocessing: The third stage of the ML pipeline is data preprocessing, which involves preparing and cleaning the dataset for training the ML model. This includes steps such as removing missing values, handling outliers, normalizing data, and encoding categorical variables. Data preprocessing is important because it helps to ensure that the data is in a suitable format for the ML algorithm and improves the accuracy of the model.

Feature Selection: Feature selection is the process of selecting the most relevant features or variables from the dataset that can help the ML model to make accurate predictions. This step is important because it can help to reduce the dimensionality of the data and improve the efficiency and accuracy of the model. Some common feature selection techniques include correlation analysis, principal component analysis (PCA), and mutual information.

Modeling Various Algorithms: After feature selection, the next step is to choose an appropriate ML algorithm for building the prediction model. There are several ML algorithms available, and the choice of algorithm depends on the problem being solved, the type of data, and the performance metrics used to evaluate the model. Some common ML algorithms used in prediction modeling include:

1. Linear regression: This algorithm is used to model linear relationships between the input features and the output variable.
2. Logistic regression: This algorithm is used for binary classification problems, where the output variable is binary.
3. Decision trees: This algorithm is used for both classification and regression problems and works by creating a tree-like model of decisions and their possible consequences.
4. Random forest: This algorithm is an ensemble learning method that combines multiple decision trees to improve the accuracy of the model.
5. Support Vector Machines (SVM): This algorithm is used for both classification and regression problems and works by creating a hyperplane that separates the data into different classes.

Hyperparameter Tuning:

- Hyperparameters are parameters that are set prior to training and control the behavior of the model.
- Hyperparameter tuning involves selecting the optimal values for these hyperparameters to minimize the error of the model on the validation set.

- Techniques such as grid search, random search, early stopping, and Bayesian optimization can be used to tune hyperparameters.
- Hyperparameter tuning can be a computationally expensive process, especially for complex models and large datasets.

Choosing the Best Model:

- Once the hyperparameters have been tuned, the performance of the model can be evaluated on a holdout test set.
- Metrics such as accuracy, precision, recall, F1 score, and AUC-ROC can be used to evaluate the performance of the model.
- The best model is the one that performs the best on the test set according to the chosen performance metric(s).
- It's important to consider the complexity of the model when selecting the best model, as more complex models may overfit the data and reduce the generalization ability of the model.
- Model interpretability is also an important consideration, especially in fields such as healthcare and finance where decisions made by the model may have significant consequences.
- It's also important to consider the scalability and computational efficiency of the model, especially when dealing with large datasets and real-time applications.

Deployment Architecture

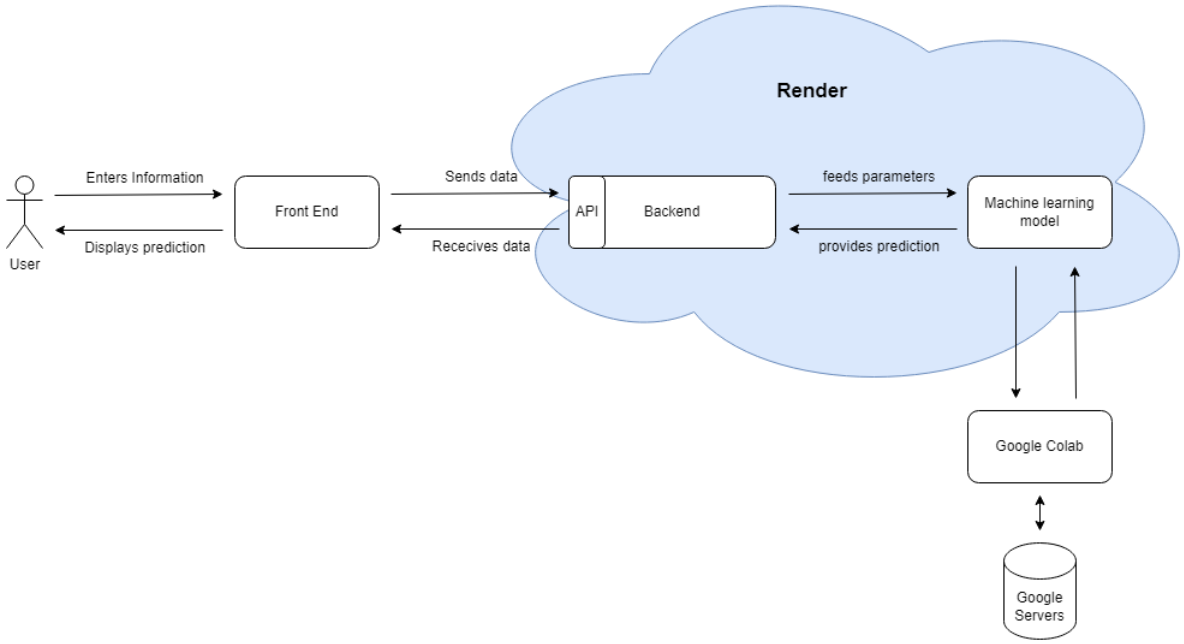


Figure 2 Building and Deployment Architecture

User

The user of a deployed machine learning model can vary depending on the specific application and use case of the model. Some examples of potential users of this deployed machine learning model:

- End-users: End-users are the individuals or organizations that use the model to make predictions or decisions. For example, a healthcare provider may use this machine learning model to predict the risk of developing PCOS in a patient, or a financial institution may use a model to predict credit risk.
- Data scientists: Data scientists may use this deployed machine learning model to evaluate the performance of the model, identify areas for improvement, and fine-tune the model for better results.
- Developers: Developers may integrate this deployed machine learning model into an application or platform, such as a mobile app or web service, to provide predictive capabilities to end-users.

But in our architecture we are only exposing our service to the End User which will consume our service via a website frontend.

Frontend

The interactive website written using web native languages- HTML, CSS and Javascript serves as primary communication between the user and the deployed model. It makes it easier for the user to query the model and get predictions without knowing the inner working of the model and just providing enough information to generate a prediction.

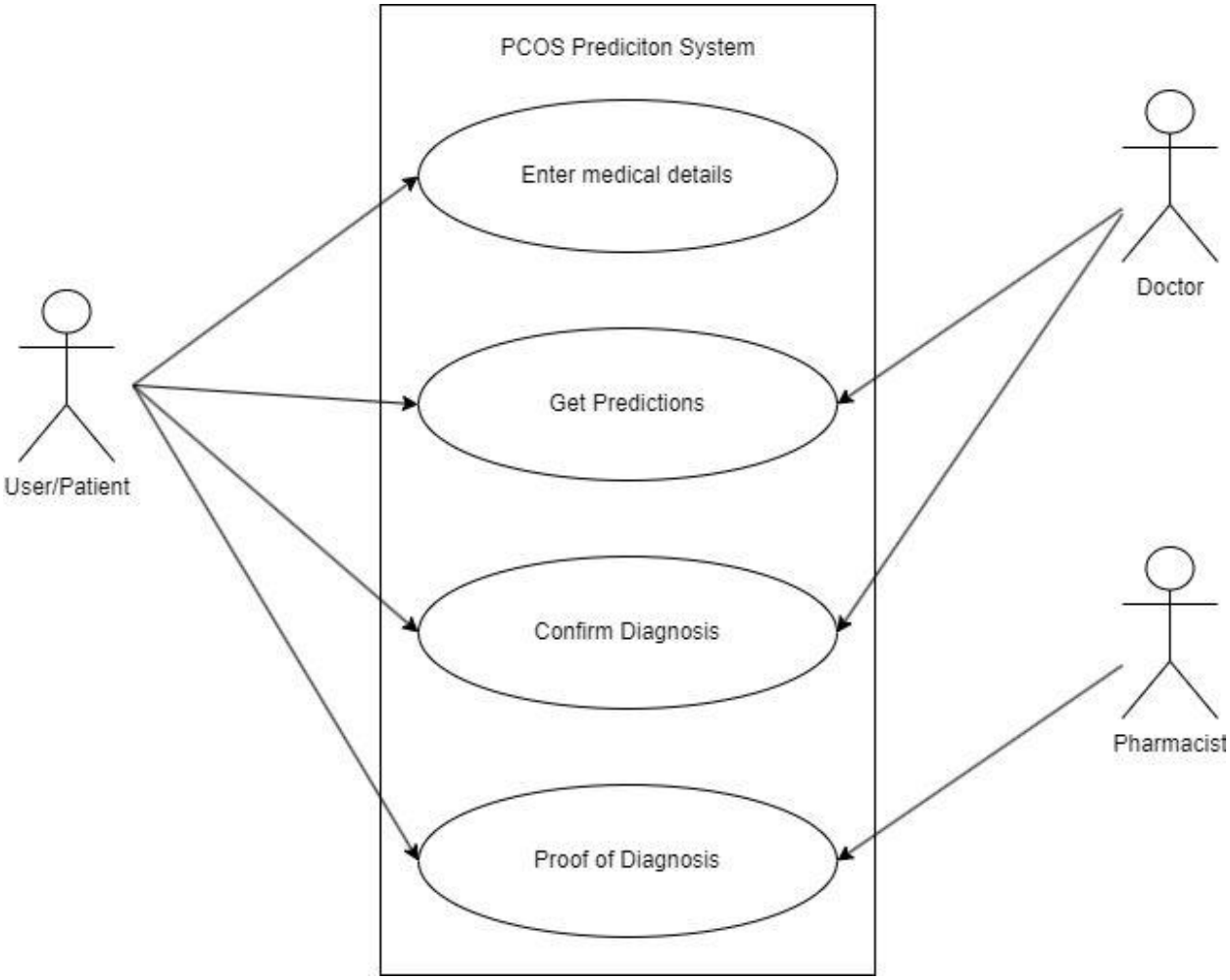
Backend API

Written in Python and deployed on Render, this API takes in the requests sent by the user via the frontend and generates predictions using the stored prediction model on the server. It is a REST API built with FAST API, a python library for building web apis. This API as mentioned is deployed on the PaaS Render.

Google Colab and Storage

Google Colab the free online on cloud Jupyter notebook is used to write the ML code. The datasets are stored in Google Drive.

Use case diagram



Implementation

The training dataset is from Kaggle –

https://www.kaggle.com/datasets/prasoonkottarathil/polycystic-ovary-syndrome-pcos

The prediction model was developed on Google Colab in Python using machine learning libraries like sklearn –

https://colab.research.google.com/drive/1ivZ_Pe6acEc6dR4_60bSRhVjUfXGhYNe

The backend was also built in Python using FastAPI, the backend code is present in the following Github repository –

https://github.com/ranmerc/pcos-prediction-backend/

And deployed on Render using Render's Native Python Runtime(https://render.com/docs/native-runtimes), deployed website present on the following link –

https://pcos-prediction-backend.onrender.com/

The frontend was built in NextJS and React with the following package dependencies:

```
  "dependencies": {
    "@emotion/react": "^11.11.0",
    "@emotion/styled": "^11.11.0",
    "@mui/material": "^5.13.2",
    "@tanstack/react-query": "^4.29.12",
    "@tanstack/react-query-devtools": "^4.29.12",
    "axios": "^1.4.0",
```

```
    "formik": "^2.3.2",
    "next": "latest",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "yup": "^1.2.0"
  },
  "devDependencies": {
    "@tanstack/eslint-plugin-query": "^4.29.9",
    "@types/node": "^18.0.0",
    "@types/react": "^18.0.14",
    "@types/react-dom": "^18.0.5",
    "typescript": "^4.7.4"
  }
```

The prediction frontend is deployed on Vercel, which provides zero configurations NextJS deployments –

https://pcos-prediction-frontend.vercel.app/

Result

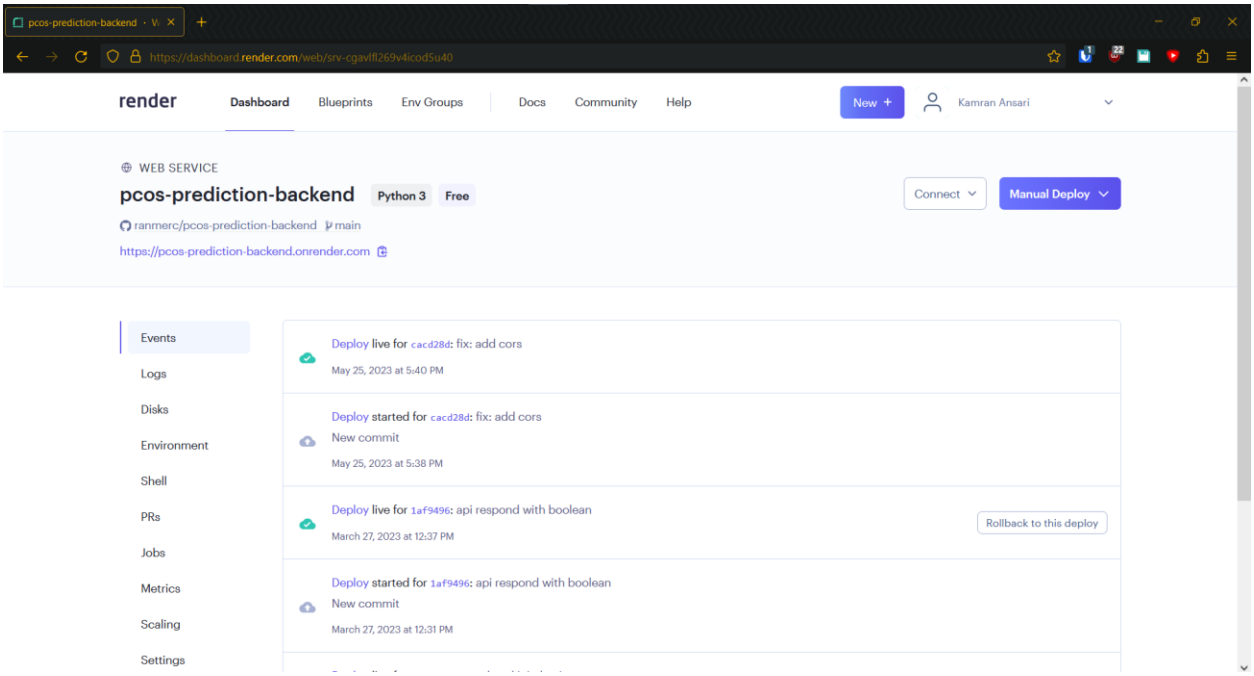


Figure 3 Backend Deployment Dashboard on Render

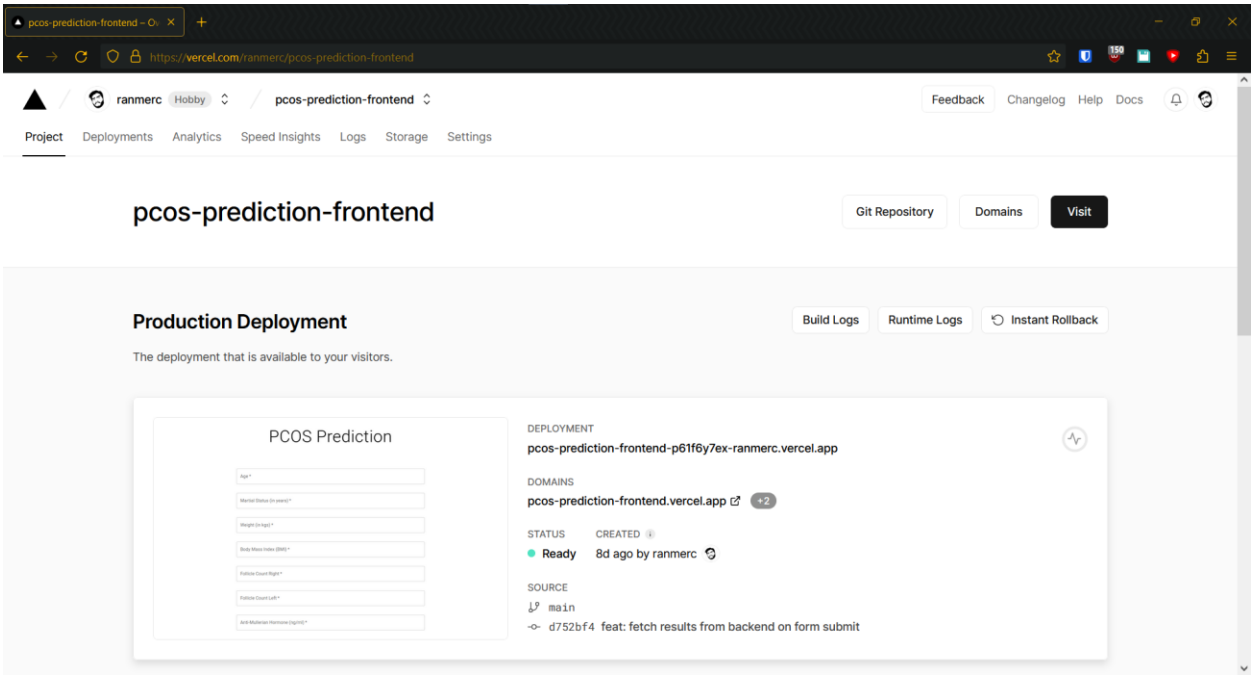


Figure 4 Frontend Deployment Dashboard on Vercel

PCOS Prediction

Age *

Martial Status (in years) *

Weight (in kgs) *

Body Mass Index (BMI) *

Follicle Count Right *

Follicle Count Left *

Anti-Mullerian Hormone (ng/ml) *

Regular Period Cycle (Length) *

Period Cycle Length (Length) *

Is your skin darkening? *

- ○ Yes
- ◉ No

Are you experiencing excessive facial hair growth? *

- ○ Yes
- ◉ No

Have you gained weight? *

- ○ Yes
- ◉ No

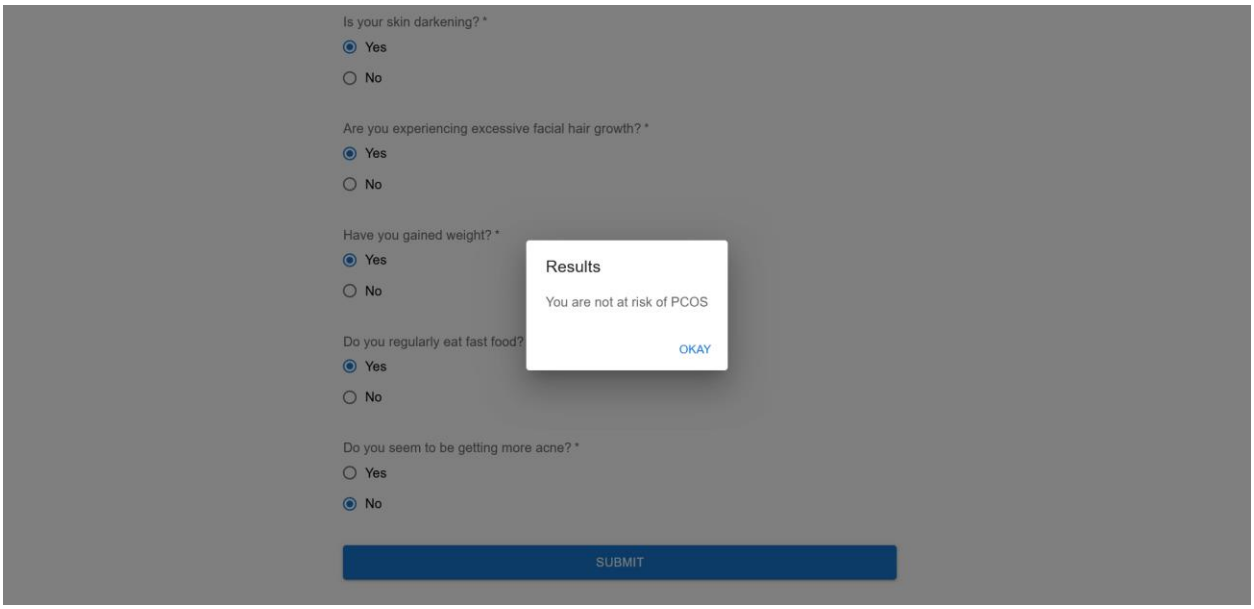
Do you regularly eat fast food? *

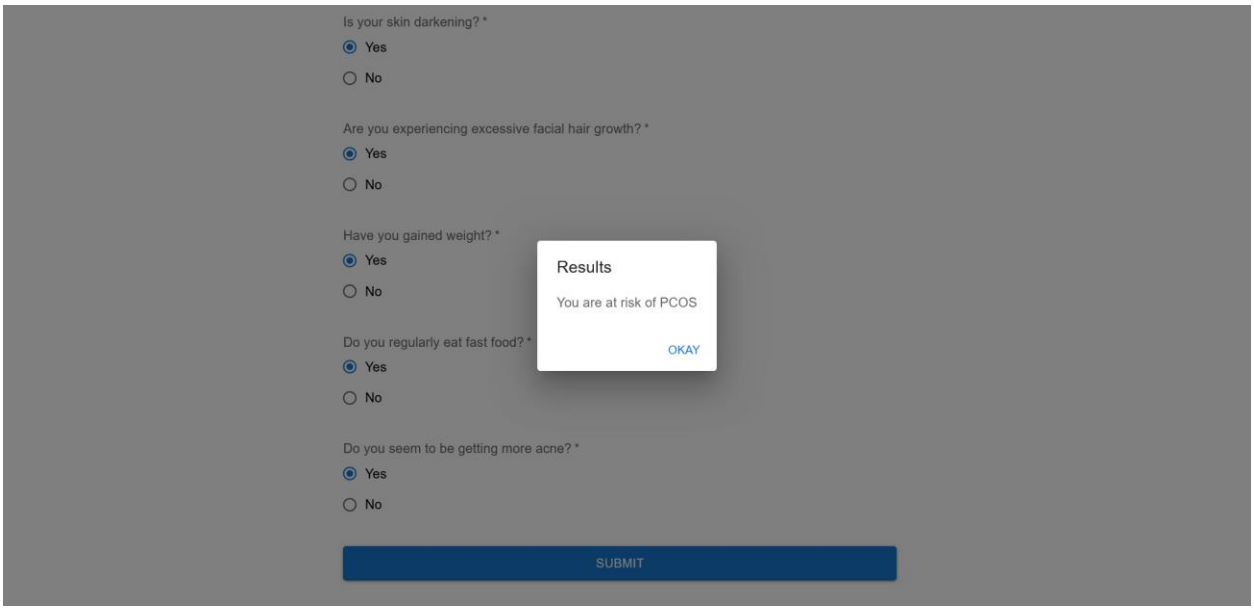
- ○ Yes
- ◉ No

Do you seem to be getting more acne? *

- ○ Yes
- ◉ No

SUBMIT


Is your skin darkening? *
Yes
No
Are you experiencing excessive facial hair growth? *
Yes
No
Have you gained weight? *
Yes
No
Results
You are not at risk of PCOS
OKAY
Do you regularly eat fast food? *
Yes
No
Do you seem to be getting more acne? *
Yes
No
SUBMIT


Is your skin darkening? *
Yes
No
Are you experiencing excessive facial hair growth? *
Yes
No
Have you gained weight? *
Yes
No
Results
You are at risk of PCOS
OKAY
Do you regularly eat fast food? *
Yes
No
Do you seem to be getting more acne? *
Yes
No
SUBMIT

Conclusion

In conclusion, this project successfully tackled the challenges of deploying a PCOS detection machine learning model on the cloud, leveraging the capabilities of big cloud providers and taking advantage of the benefits offered by SaaS platforms.

The backend of the PCOS detection system was hosted on Render, providing a robust infrastructure that ensured scalability, performance, and security. The backend, accessible at https://pcos-prediction-backend.onrender.com/, efficiently processed incoming requests, executed the machine learning model, and delivered accurate PCOS predictions to the frontend.

The frontend website, responsible for collecting user data for PCOS prediction, was hosted on Vercel at https://pcos-prediction-frontend.vercel.app/. This website provided an intuitive and user-friendly interface, allowing individuals to input their relevant information, such as medical history, symptoms, and clinical attributes. The data collected was securely transmitted to the backend for analysis and prediction.

By combining the power of Render and Vercel, this project achieved a seamless integration between the backend and frontend components, ensuring smooth data flow and enabling an efficient PCOS prediction process. The deployment on these SaaS platforms simplified infrastructure management, provided scalability, optimized costs, and delivered robust security measures, ensuring the privacy and confidentiality of patient data.

With the successful deployment of the PCOS detection model on the cloud, this project contributes to the advancement of healthcare diagnostics. It offers a standardized, accurate, and accessible solution for PCOS detection, empowering healthcare providers to diagnose PCOS more efficiently and enabling early intervention to mitigate long-term health risks. Additionally, the user-friendly frontend website enhances the overall user experience, making it convenient for individuals to assess their PCOS risk.

The combination of Render and Vercel as the hosting platforms for the backend and frontend respectively showcases the capabilities of these SaaS platforms in facilitating the deployment and management of machine learning models. This successful implementation opens up opportunities for further exploration and enhancement in the field of cloud-based healthcare diagnostics.

In summary, the deployment of the PCOS detection model on Render and Vercel marks a significant milestone in improving PCOS diagnostics. By offering an accurate, accessible, and user-friendly solution, this project contributes to the overall well-being of women affected by PCOS and paves the way for future advancements in leveraging machine learning in healthcare.

References

- "A Comparative Study of Performance of Cloud-Based Web Hosting Providers", A. Yeluri, D. Patnaik, R. Reddy
- "Efficient Resource Management for Cloud-Hosted Web Applications", X. Meng, W. Shi, J. Qiu, H. Mei
- A Systematic Literature Review on Cloud Computing Security: Threats and Mitigation Strategies
- "Above the Clouds: A Berkeley View of Cloud Computing" by Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia
- "A Survey of Cloud Computing Architecture and Applications" by Akshay Kumar, Neha Jain, and Bhawna Mallick
- "Cloud Computing Security: From Single to Multi-Clouds" by Siani Pearson and Yogendra Shah
- S. Prapty and T. T. Shitu, “An Efficient Decision Tree establishment and performance analysis on Polycystic Ovary Syndrome”
- S. Bharati, P. Poddar and M. R. Hossain Mondal, “Diagnosis of Polycystic Ovary Syndrome using Machine Learning Algorithms”
- Y. A. Abu Adla, D. G. Raydan, M. Z J. Charaf, R. A. Saad, J. masreddine and M. O. Diab, “Automated Detection of Polycystic Ovary Syndrome using Machine Learning”
- “Deploying Machine Learning Models on Cloud: A Survey" by S. K. Gouda et al. (2020)
- "Container-Based Deployment of Machine Learning Models: A Case Study" by M. A. Khan et al. (2021)
- "Challenges and Best Practices for Deploying Machine Learning Models on Cloud" by S. Banerjee et al. (2021)
- FastAPI official tutorial - https://fastapi.tiangolo.com/tutorial/
- Python Tutorial: VENV (Windows) - How to Use Virtual Environments with the Built-In venv Module by Cory Schafer on Youtube
- Deploying FastAPI application to Render by Akash R Chandran
- What and why behind fit_transform() and transform() in scikit-learn! by Towards Data Science
- React Query Docs - https://tanstack.com/query/v4/docs/react/quick-start
- MUI Docs - https://mui.com/material-ui/getting-started/overview/
- Yup Docs - https://github.com/jquense/yup
- How to input decimal places in Textfield from MUI in react on StackOverflow.
- Ensure that array only contains set values from a variable, issue on yup's github.
- Status of useMutation not shared across usages/components with custom hook (like useQuery does), issue on tanstack/query github.