**Plan B (Independent Study) Project Report**

**Project Title:**

Predicting Likelihood of Cardiac Health

**Student Name:**

Uma Sankara Sai Ganta Venkata

**Course Information:**

* **Course:** STA 698 - Plan B Project (3 credits)
* **Semester:** Spring 2025
* **Instructor:** Patrick Kinnicutt

**Project Summary**

This project focuses on building a complete machine learning pipeline to predict the 10-year risk of coronary heart disease (CHD). By utilizing the Framingham Heart Study dataset, we designed multiple machine learning models, integrated them within an R Plumber API, developed a web frontend using Flask, containerized the solution with Docker, and successfully deployed the entire system to Google Cloud Run. The result is a scalable, cloud-based cardiac risk prediction application accessible in real-time.

**Key Objectives**

* Perform preprocessing and cleaning of the Framingham Heart Study dataset.
* Train and evaluate multiple machine learning models: Logistic Regression, Ridge Regression, Random Forest, and XGBoost.
* Create an R Plumber API for serving model predictions.
* Build a user-friendly frontend application with Flask and Chart.js.
* Containerize the entire project using Docker.
* Deploy the containerized application to Google Cloud Run.

**Step 1: Dataset and Preprocessing**

* **Dataset Used:** Framingham Heart Study dataset.
* **Preprocessing Activities:**
  + Imputed missing numerical values using mean imputation.
  + Imputed missing categorical values using mode.
  + Converted necessary variables into factor types to maintain consistency.
  + Conducted correlation analysis and visualized relationships among variables to assist model feature selection.

**Step 2: Model Building**

* **Machine Learning Models Developed:**
  + Logistic Regression: Provided baseline performance.
  + Ridge Regression: Applied L2 regularization to address multicollinearity.
  + Random Forest: Robust ensemble learning technique.
  + XGBoost: High-performing gradient boosting model.

**Step 3: Model Evaluation**

* **Evaluation Metric:** ROC-AUC Score.
* **Performance Results:**
  + Logistic Regression AUC: **0.727**
  + Ridge Regression AUC: **0.7255**
  + Random Forest AUC: **1.0**
  + XGBoost AUC: **0.9993**

**Step 4: API Development (R Plumber)**

* Developed api.R exposing the /predict endpoint.
* Built entrypoint.R to load models (logistic, ridge, random forest, XGBoost) during container startup.
* Ensured correct factor level matching for incoming API requests.

**Step 5: Web Application (Flask Frontend)**

* app.py was designed to:
  + Collect user input through a modern Bootstrap-based form.
  + Send collected data to the R Plumber API.
  + Display model predictions with percentages.

**Step 6: Dockerization**

* **Dockerfile:**
  + Base image: rocker/r-ver:4.3.1.
  + Installed all required R packages and Python packages listed in requirements.txt.
  + Defined /app as working directory and ensured proper permissions on scripts.
* **entrypoint.sh:**
  + Launched R API and Flask App together.
  + Ensured both servers started concurrently and handled lifecycle gracefully.

**Step 7: Deployment to Google Cloud Run**

* Built and tagged the Docker image locally.
* Pushed image to **Artifact Registry** under project **heart-risk-app**.
* Deployed the container to **Google Cloud Run** in **us-central1** region.
* Configured the service for public access without requiring authentication.

**Live URL:** [Heart Risk App](https://heart-risk-app-802586937044.us-central1.run.app/)

**File Structure Overview**

* **api.R** — R script defining the API.
* **entrypoint.R** — R script for model loading and API launch.
* **entrypoint.sh** — Bash script starting both servers.
* **Dockerfile** — Instructions to build the Docker image.
* **app.py** — Python Flask application.
* **requirements.txt** — Python dependency list.
* **framingham.xlsx** — Health dataset used.

**Challenges Faced**

* **Docker Communication:** Solved R API and Flask server communication inside Docker.
* **Model Loading Issues:** Addressed timing and path problems while loading models.
* **Input Validation:** Resolved type mismatch (string vs numeric) during prediction requests.
* **Deployment Hurdles:** Fine-tuned container startup timing to meet Cloud Run health checks.

**Conclusion**

The project successfully resulted in a deployable, scalable, and easy-to-use heart disease risk prediction system. It not only integrated statistical modeling techniques but also full-stack development skills including cloud deployment.

**Future Enhancements**

* Implement user login and authentication to secure access.
* Strengthen input validation and error handling.
* Integrate explainable AI techniques (like SHAP) to interpret model decisions.