Breast Cancer Diagnosis & Analysis

Amanda Lee
Email: amanda.lee@utexas.edu, EID: awl646

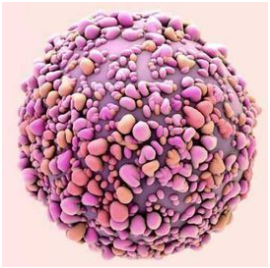
Dhanny Indrakusuma,
Email: dhannywi@utexas.edu EID: dwi67

Pranjal Adhikari
Email: pranjal.adhikari@utexas.edu EID: pa8729

Spring 2023

Contents

1 Introduction



Beginning at the University of Wisconsin-Madison, Dr. William H. Wolberg in the Department of Surgery and Human Oncology researched machine learning approaches to apply breast cancer diagnosis and prognosis. Using nine visually assessed characteristics of a Fine Needle Aspiration (FNA) sample, Dr. Wolberg could accurately diagnose breast masses. With the help of Professor Olvi L. Mangasarian (and two of his graduate students, Rudy Setiono and Kritin Bennett) in the Department of Computer Sciences, their collaboration constructed a classifier that uses the multi-surface method (MSM) of pattern separation on those nine features to diagnose 97% of new breast cancer cases successfully. The resulting data set became known as the Wisconsin Breast Cancer Data. The image analysis work began in 1990 (with the addition of Nick Street) to diagnose samples with a digital image of a small section of the FNA slide, which the research team consolidated into a software system called *Xcyt*.

This is the diagnosis process:

- An FNA is taken from the breast mass, mounted on a microscope slide, and stained to highlight the cellular nuclei.
- The user then isolates the individual nuclei with *Xcyt* by drawing the approximate boundary of each nucleus and using computer vision approaches ("snakes") to converge to exact nuclear boundaries. This process takes about two to five minutes per slide.
- Once all of the nuclei have been isolated, the program computes values for the ten characteristics of the nuclei (shape, size, texture, etc.).
- From 569 cases, a classifier was constructed to differentiate between benign and malignant samples. It allows for the computation of the probability of malignancy in new patients.

1.1 Breast Cancer Data Analysis

This is the attribute information in the Wisconsin Breast Cancer Data set that we will focus on:

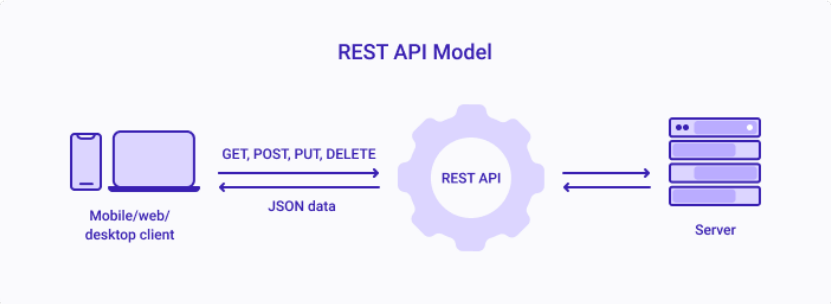
1. ID number
2. Diagnosis (M = Malignant, B = Benign)
3. Radius (mean of distances from the center to points on the perimeter)
4. Texture (standard deviation of grey-scale values)
5. Perimeter
6. Area
7. Smoothness (local variation in radius lengths)

8. Compactness ($\frac{Perimeter^2}{Area} - 1.0$)
9. Concavity (severity of concave portions of the contour)
10. Concave Points (the number of concave portions of the contour)
11. Symmetry
12. Fractal Dimension ("$CoastlineApproximation$" $- 1$)

2 Diagnosing Breast Cancer with a REST API

From the benign/malignant breast cancer data set, we built a REST API with **Python** to perform the same ability to differentiate between benign and malignant cases. Utilizing the ten characteristics in the data set, an effective and reliable diagnosis can be made to determine each case's outcome and predict the survival status based on these features.

2.1 What is a REST API?

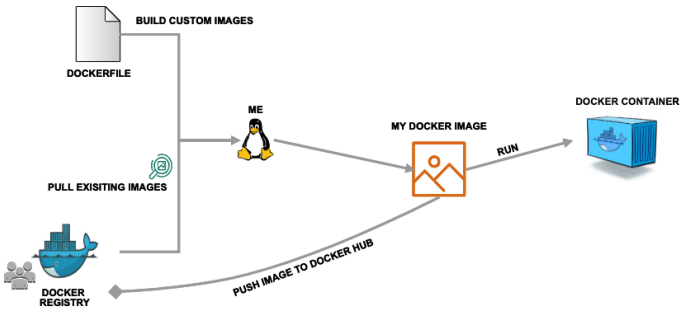


A **REST API** (or RESTful API) is an application programming interface (API or web API) that conforms to the constraints of REST architectural style (defines a set of rules) and allows interactions with RESTful web services. APIs are useful for allowing larger software systems to be built from smaller components, allowing the same component or code to be used by different systems, and insulating consumers from changes to the implementation. They provide the functionality to external software in the form of a contract specifying the inputs the consuming software must provide and the outputs that the API will produce from the inputs and conceal the implementation of this functionality from the consuming software so that changes can be made to the implementation without impacting consumers. Additionally, APIs provide errors when the consuming software doesn't fulfill the contract of the API or when unexpected circumstances are encountered. We also are using Flask, Redis, Docker, and Kubernetes with this REST API.

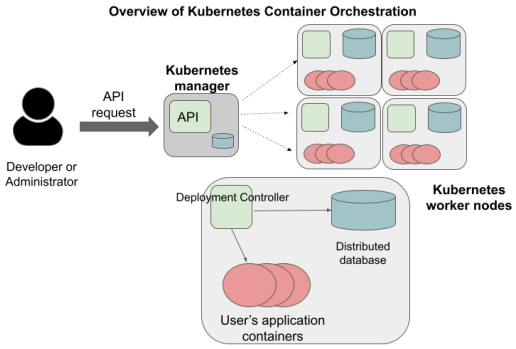
Flask is a Python library and framework for building web services that contribute to the modularity, abstraction, and generalization of software. It helps define and implement various "routes" or API endpoints in a Flask Python program for users to retrieve data.

Redis is a NoSQL (No Structured Query Language) database and "data structure store". It provides key-value store functionality (stored `key:value` objects). Redis helps separate the database service from the application container (modularity design principle), extending the life of the application (or user) data and gaining the ability to access it from outside the container.

Docker is a containerization platform that allows users to package an application and its dependencies in deliverable units (containers); isolate it from other applications and services; and deploy it consistently, reproducibly, and platform-agnostically (contributing the the portability of software projects).



Kubernetes (k8s) is a distributed system of software components that run a cluster of one or more machines (physical computers or virtual machines). Each machine in a Kubernetes cluster is either a "manager" or "worker" node. Communication with Kubernetes can be done when users make requests to its API. Overall, Kubernetes improves software portability, especially for large distributed applications that need to run across multiple machines.



We implemented a **Jobs API** as an asynchronous endpoint to perform a long-running task. Instead of performing the actual computation, the Jobs API will record that the user has requested that computation, store it in Redis, and immediately respond to the user. The response doesn't include the job's result, but it will indicate that the request was received and will be worked on in time. Also, it will provide an id for the job that the user can use to check the status and the (later) actual result.

2.2 Route Endpoints

The following routes and methods are used to query for the data.

```
\data -X POST
Loads Wisconsin Breast Cancer Data into a Redis database

\data -X GET
Returns Wisconsin Breast Cancer Data from the Redis database

\data -X DELETE
Deletes Wisconsin Breast Cancer Data in the Redis database

/id
Returns a JSON-formatted list of all ID Numbers
```

```
/id/<id_num>
Returns all breast cancer data (ten characteristics) associated with ID
↪  Number <id_num>

/outcome
Returns a JSON dictionary containing information regarding benign and
↪  malignant cases

/diagnosis-mean-radius
Returns a JSON dictionary with the mean redius information based on breas
↪  cancer diagnoses

/image -X POST
Creates a plot of the number of benign and malignant cases and saves it to
↪  the Redis database

/image -X GET
Returns a plot image of the number of benign and malignant cases to the user
↪  if present in the Redis database

/image -X DELETE
Deletes the plot image of the number of benign and malignant cases from the
↪  Redis database

/jobs -X POST...
Submits a job to the worker for data analysis

/jobs/<job_id>
Returns the status of the <job_id>

/download/<job_id>
Returns the plot associated with <job_id>

/help
Returns a help text that briefly describes each route
```

2.3 Implementation and Example Outputs of Routes

To query the Flask application, users of the API insert the following on the command line (depending on what information a user would like to see):
1. To return a help text that briefly describes each route:

```
$ curl localhost:5000/help

Usage: curl [host_name]:5000[ROUTE]
A Flask REST API for querying and returning interesting information from the
breast cancer diagnosis dataset.
Route                   Method  What it returns
/data                   POST    Put data into Redis database
/data                   GET     Return entire data set from Redis database
/data                   DELETE  Delete data in Redis database
/id                     GET     Return json-formatted list of all "ID Number"
/id/<id_num>            GET     Return all data associated with <id_num>
/outcome                GET     Return a dictionary with information on malignant
```

```
                                  and benign cases with associated ID Numbers
/diagnosis-mean-radius  GET       Return a dictionary with Mean Radius information
                                  based on diagnosis
/image                  POST      Creates a plot and saves it to Redis
/image                  GET       Returns the plot created
/image                  DELETE    Delete the plot saved in Redis database
/jobs                   POST      Submits job to worker for analysis of data
/jobs/<job_id>          GET       Returns the status of the <job_id>
/download/<job_id>      GET       Returns the plot associated with <job_id>
/help                   GET       Return help text that briefly describes each route
```

2. To load the breast cancer data to a Redis database:

```
$ curl localhost:5000/data -X POST
Data loaded in
```

3. To return all data from the Redis database:

```
$ curl localhost:5000/data -X GET
[
.
    },
        {
            "Area SE": "156.8",
            .
            .
            "Worst Texture": "28.18"
        },
    {
.
]
```

4. To delete the breast cancer data to a Redis database:

```
$ curl localhost:5000/data -X DELETE
Data deleted
```

5. To access a list of all ID Numbers of breast cancer patients:

```
$ curl localhost:5000/id
[
.
.
    "91979701",
    "901549",
    "905520",
.
.
]
```

6. To return information about a certain patient (via their ID Number):

```
$ curl localhost:5000/id/<id_num>
{
  "Area SE": "38.49",
  .
  .
  "Worst Texture": "28.06"
}
```

7. To return a dictionary of information about benign and malignant breast cancer cases with an associated ID Number:

```
$ curl localhost:5000/outcome
{
  "Benign": {
    "IDs": [
      "918192",
      .
      .
      .
      "869224"
    ],
    "Total cases": 357
  },
  "Malignant": {
    "IDs": [
      "852781",
      .
      .
      .
      "866083"
    ],
    "Total cases": 212
  }
}
```

8. To return a dictionary with the mean radius based on diagnoses:

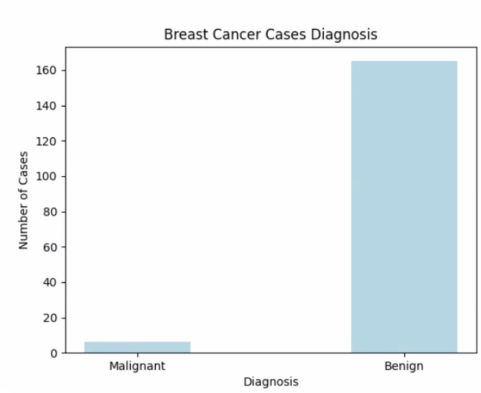
```
$ curl localhost:5000/diagnosis-mean-radius
{
  "Benign": {
    "cases": 357,
    "mean_radius": {
      "avg": 12.15,
      "max": 17.85,
      "min": 6.981
    }
  },
  "Malignant": {
    "cases": 212,
    "mean_radius": {
      "avg": 17.46,
      "max": 28.11,
      "min": 10.95
    }
  }
}
```

9. To create a plot and load it to the Redis database:

```
$ curl localhost:5000/image -X POST
Graph successfully saved
```

10. To return the plot image from the Redis database:

```
$ curl localhost:5000/image -X GET --output image.png
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 19318  100 19318    0     0  1257k      0 --:--:-- --:--:-- --:--:-- 1257k
```



11. To delete the plot image from the Redis database:

```
$ curl localhost:5000/image -X DELETE
Image deleted
```

12. To submit a job to a worker for breast cancer data analysis:

```
$ curl localhost:5000/jobs -X POST -d '{"start":6, "end":12}' -H "Content-Type:
↪  application/json"
{
  "id": "9905ff66-c92b-4e01-a455-a847af81b31d",
  "status": "submitted",
  "start": 6,
  "end": 12
}
```

13. To return the status of the job ID:

```
$ curl localhost:5000/jobs/9905ff66-c92b-4e01-a455-a847af81b31d
{
  "id": "9905ff66-c92b-4e01-a455-a847af81b31d",
  "status": "in progress",
  "start": "6",
  "end": "12"
}
```

3 Ethical and Professional Responsibilities

As a developer creating a REST API for analyzing breast cancer, there are ethical and professional responsibilities to consider, which include the following:

1. You must ensure patient confidentiality and privacy by taking any necessary measures to protect personal health information. This includes securing the data in transit and at rest and ensuring that only authorized personnel can access the data.

2. You must provide accurate and reliable information, meaning the data provided through the API (the Wisconsin Breast Cancer Data) is accurate and reliable.

3. You must avoid any biases or discrimination (i.e. the use of language or data that could ensue stereotypes or contribute to discrimination), especially in the data provided through the API.

4. You must allow access to all users, including those with disabilities.

5. You must comply with legal and ethical standards, including all legal and ethical standards such as patient privacy and data protection.

In conclusion, as a developer creating a REST API for breast cancer analysis, there is a responsibility to ensure patient confidentiality, provide accurate information, avoid biases and discrimination, allow accessibility, and comply with legal and ethical standards. Ensuring these responsibilities will help promote the ethical and responsible uses of technology in the biomedical and healthcare industry.

4 Bibliography

"Machine Learning for Cancer Diagnosis and Prognosis.",
pages.cs.wisc.edu/ olvi/uwmp/cancer.html

"Breast Cancer Wisconsin (Prognostic) Data Set."
UCI Machine Learning Repository,
archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Prognostic).

"COE 332: Software Engineering Design.",
coe-332-sp23.readthedocs.io/en/latest/index.html.