Scientific Programming Final Project

Breast Cancer Diagnosis API

Authors:

Alejandro Cabello Bonal, Ernest Ceballos Ortega, Júlia Galimany Claver, Oriol Galimany Garriga, Noa-Sibila Janer Oliver, Iván Pérez López and Nerea Salvador Prieto.

Date: January 2026

GitHub Repository:

https://github.com/Blanqui04/Scientific-Programming-Final-Project---Group-C/tree/main

---

1. Project Objective

The aim of this project is to deploy a reliable and accurate predictive model that is able to predict diagnoses when prompted with specific features from breast cancer patient measures, in this case computed using images of Fine Needle Aspirate (FNA) slides [1]. The resulting model will be encapsulated in an API and containerized with Docker, allowing users to input markers for a given subject and receive a possible diagnosis in a portable computing environment.

---

2. Task Distribution and Team Contributions

The tasks were randomly assigned among team members following the task division proposed in the project description. The assignments are as follows:

Task description	Input / Output	Contributor
Create a function to rename DataFrame columns using dict_names. Summarize cleaned data and annotate observations (e.g., outliers, variability).	**Inputs**: df, dict_names **Outputs**: renamed_df, summary notes	Nerea Salvador Prieto

- Create scatterplot function for two variables grouped by category.	**Inputs**: cleaned_dataframe, var1, var2, groups **Outputs**: Aggregated DataFrame, scatterplot	Iván Pérez López
- Normalize all variables in df using z-score or min-max scaling (justify choice). - Analyze correlations among fundamental	**Inputs**: Original DataFrame **Outputs**: Normalized DataFrame, cleaned_dffrequency, Jitter, and Shimmer variables; keep representative ones and remove others.	Oriol Galimany Garriga
- Write function to aggregate variables by grouping variable using pandas groupby. - Implement classification (e.g., KNN or other model) to distinguish patients vs. controls.	**Inputs**: df, gv **Outputs**: Aggregated DataFrame, classification results	Ernest Ceballos Ortega
- Implement validation strategy (train/test split or cross-validation). - Select best model for API implementation based on validation results.	**Inputs**: Training and test sets **Outputs**: Validation metrics, selected model	Alejandro Cabello Bonal
- Design and implement API using FastAPI or Plumber. - Define endpoints for input markers and prediction output. - Integrate trained model into API and add documentation/tests.	**Inputs**: Final model, feature schema **Outputs**: Functional API with documentation	Júlia Galimany Claver
- Containerize API using Docker (create Dockerfile). - Deploy API on a server (cloud or local). - Test deployed API and document deployment steps.	**Inputs**: API code, Docker configuration **Outputs**: Docker image, running API on server	Noa-Sibila Janer Oliver
Improve and select the best	**Inputs**: Several different	All

predictive model, write the report and record the video.	models, API **Output**: Report and video of your work	

---

3. Analysis and Key Results

3.1 Data Analysis and Treatment

The first step was to read the dataset and analyse its features. The Breast Cancer Wisconsin dataset contains only numeric values, even for the diagnosis attribute, which is a binary (0 for malignant, 1 for benign outcome). These variables are measures like cell nuclei radius (average radius), perimeter, area, texture (standard deviation of gray-scale values), smoothness (local variation in radius lengths), compactness (perimeter^2/area), concavity (severity of concave portions of the contour), number of concavities, symmetry and fractal dimension (approximated using the “coastline approximation”). For each of these measures, there is an additional attribute that measures its standard error and another attribute that stores the worst case computed on that particular nucleus.

Whitespaces in the names were replaced by underscores for practicality, and no NA values could be found, so we proceeded to the analysis. There was a selection of several combinations of different variables, which were plotted (Fig. X) using different colors for each diagnosis category (blue for benign, red for malignant) to get a first overview of what attributes may be useful in distinguishing diagnostic clusters, and thus useful for prediction. Some of them were radius, area, compactness, smoothness and symmetry.

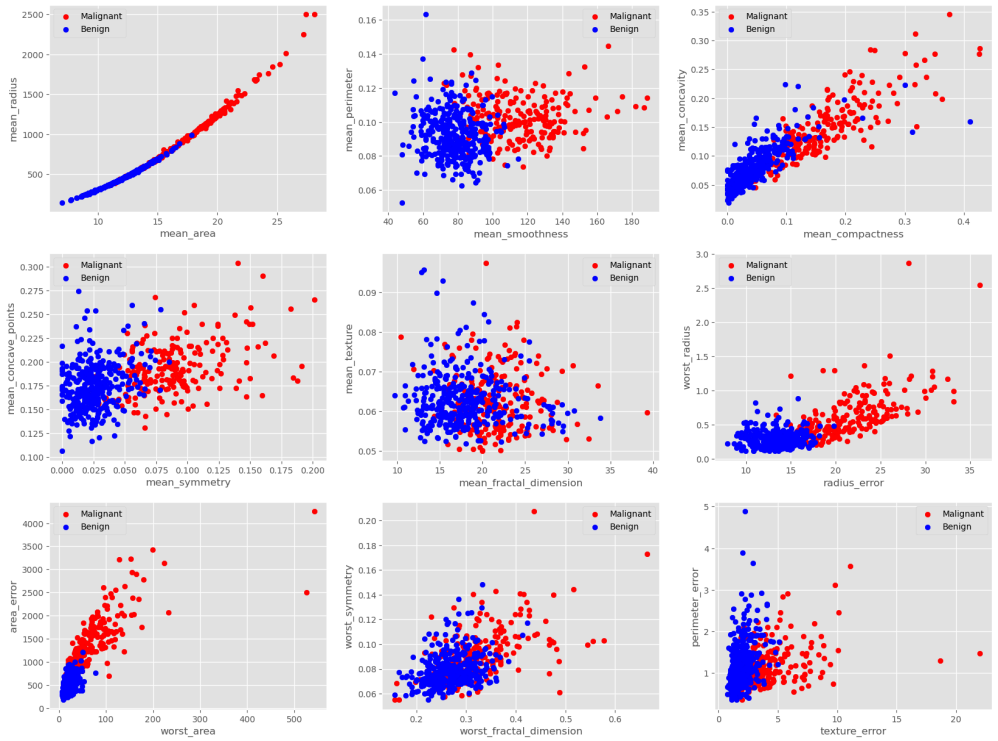

Figure 1. Scatter plots of different combinations of attributes, colored by diagnosis.

Variables were normalized using z-score to facilitate comparison. We computed a correlation matrix (Fig. X) to filter variables that were highly correlated and thus redundant for the training of a predictive model. Attributes with correlation higher than 0.9 were discarded.

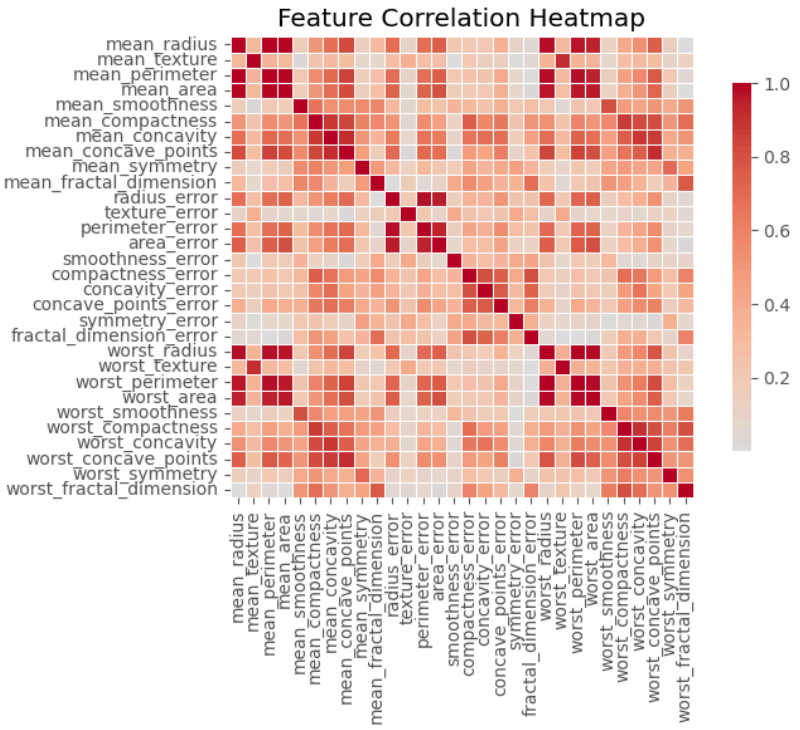

Figure 2. Correlation heatmap of all the variables in the dataset, excluding diagnosis.

3.2 Model Development and Performance

The first model implemented was K-Nearest Neighbors. We split data into 80% for training and 20% for testing. Then, we fitted the model with different values of k to select the one that yielded better performance. We set the model to k=5, which gave the best accuracy and ROC-AUC score while keeping the model simple. We also arranged the predictions in a confusion matrix (Fig. 3), and computed predictive performance metrics such as precision (proportion of true positives), recall (proportion of true negatives) and F1-score. In all cases, they were at least 0.95, ensuring validity of predictions for most cases.

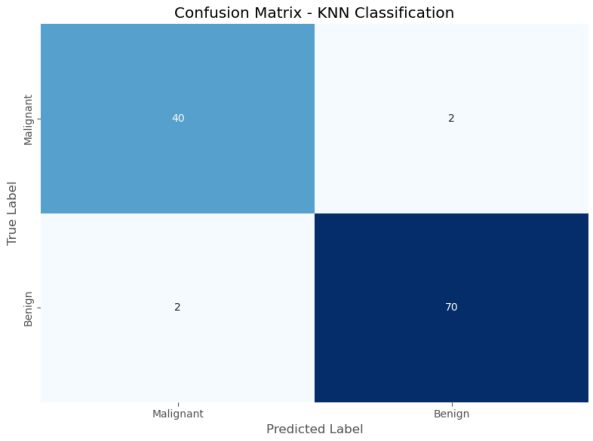

Figure 3. Confusion matrix for the predictions made by the KNN model

We implemented a decision tree classification model to compare its performance with the previous one. The split percentages were kept the same, and we repeated the same procedure for the assessment. In this case, k value is replaced by depth value, which is the number of ramifications our tree model has. The best accuracy was obtained with a maximum depth of 4. However, its performance metrics were somewhat lower, around 0.90, which placed KNN as the best model out of the two tested.

---

4. API Deployment

The trained predictive model was deployed as a REST API to allow users to obtain predictions by providing the required input features. The API was implemented using FastAPI and containerized with Docker to ensure reproducibility.

The API can be run locally by building the Docker image and executing the container, which exposes the service on port 8000. Once running, the API can be accessed via a web browser or HTTP requests. An interactive interface is available to test the prediction endpoint.

The correct deployment and usage of the API were tested locally and demonstrated in a short video.

---

5. Conclusions

In this project, we developed a pipeline to predict breast cancer diagnosis. The final pipeline is delivered as a portable REST API, containerized with Docker to ensure easy reproducibility. The dataset used for training and testing required minimal cleaning, as it contained no missing or out-of-bound values; moreover, feature processing included z-score normalization followed by correlation-based filtering to remove redundant variables.

Finally, as mentioned above, we concluded that the KNN model provided the strongest overall performance using an 80/20 train-test split and k=5.

6. Bibliography

[1] W. N. Street, W. H. Wolberg, and O. L. Mangasarian, “Nuclear feature extraction for breast tumor diagnosis” *Proceedings of SPIE, the International Society for Optical Engineering/Proceedings of SPIE*, Jul. 1993, doi: 10.1117/12.148698.