**Predicting Breast Cancer Malignancy: A Problog-based Approach with GUI Interface**

Advanced Topics in Artificial Intelligence

Prof. Emmanouil Marakakis

By Anna Manucharyan

26 June, 2023

**Table of Contents**

1. Project Description
2. Dataset Description
3. Bayesian Network
4. Architecture
5. Code Report

4.1 Importing the required libraries and modules

4.2 Defining global thresholds

4.3 Reading and preprocessing the dataset

4.4 Defining functions

4.5 Calling Problog

4.6 GUI creation

4.7 Event handling

4.8 Main program execution

4.9 Program demonstration

1. Evaluation metrics

5.1 Accuracy

5.2 Precision

5.3 Recall

5.4 F1 Score

5.5 Demonstration of the evaluation metrics window

1. Conclusion
2. References

## **Project Description:**

The aim of this project is to predict whether a person has malignant or benign breast cancer based on various parameters. The parameters used for prediction include radius, texture, perimeter, area, smoothness, compactness, concavity, symmetry, and fractal dimension of the tumor. The learning is done using the Wisconsin Breast Cancer dataset. When a new piece of data is inputted from the user, the probability of having a malignant cancer is calculated.

## **Dataset Description**

The dataset used in this project is the Wisconsin breast cancer dataset, which can be obtained from the following source: [Wisconsin Breast Cancer Dataset](https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data). This dataset contains information about breast cancer patients, including various attributes measured from images of their cell nuclei.

The dataset consists of the following columns:

- id: The unique ID assigned to each patient (not used for classification)

- diagnosis: The diagnosis indicating whether the tumor is malignant (M) or benign (B)

- radius\_mean: Mean of distances from the center to points on the perimeter

- texture\_mean: Standard deviation of gray-scale values

- perimeter\_mean: Mean size of the core tumor

- area\_mean: Mean area of the tumor

- smoothness\_mean: Mean of local variation in radius lengths

- compactness\_mean: Mean of perimeter^2 / area - 1.0

- concavity\_mean: Mean of severity of concave portions of the contour

- symmetry\_mean: Mean symmetry of cell nuclei

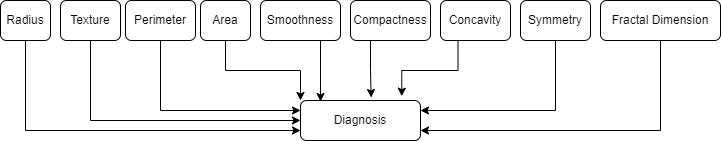
- fractal\_dimension\_mean: Mean "coastline approximation" - 1

The dataset contains many other parameters, like the standard error, worst case value of the same parameters described above. However, these parameters were not useful for my project, and even in some cases would result in an algorithm which will generalize badly, so I disregarded them.

After initial preprocessing of the data the dataset contains 9 parameters which all are mean values: radius, texture, perimeter, area, smoothness, compactness, concavity, symmetry, fractal dimension. Then the dataset is preprocessed by removing the 'id' column as it is not relevant for classification. Rows with missing values are also dropped to ensure data integrity.

## **Bayesian Network**

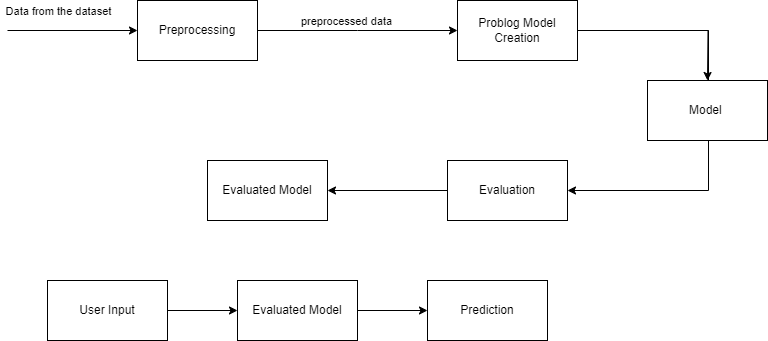
A bayesian net representing the relationships between the parameters is presented below.



All the parameters influence Diagnosis directly, since all the parameters are basically details regarding the tumor.

## **Architecture**

The provided code utilizes the Tkinter library[1] to create a GUI-based application for predicting the probability of having malignant breast cancer. The code follows the following architecture:



## **Code report**

The following steps are implemented in the code:

5.1. Importing the required libraries and modules: The code imports necessary libraries such as Tkinter[1], Pandas, and Problog for GUI, data manipulation, and probabilistic logic programming, respectively.

5.2. Defining global thresholds: The code defines a global dictionary named 'thresholds' which contains the maximum values for each parameter. These thresholds are used to categorize the input values as 'True' or 'False' based on the corresponding thresholds.

5.3. Reading and preprocessing the dataset: The code reads the Wisconsin breast cancer dataset from a CSV file. It drops the 'id' column and removes rows with missing values.

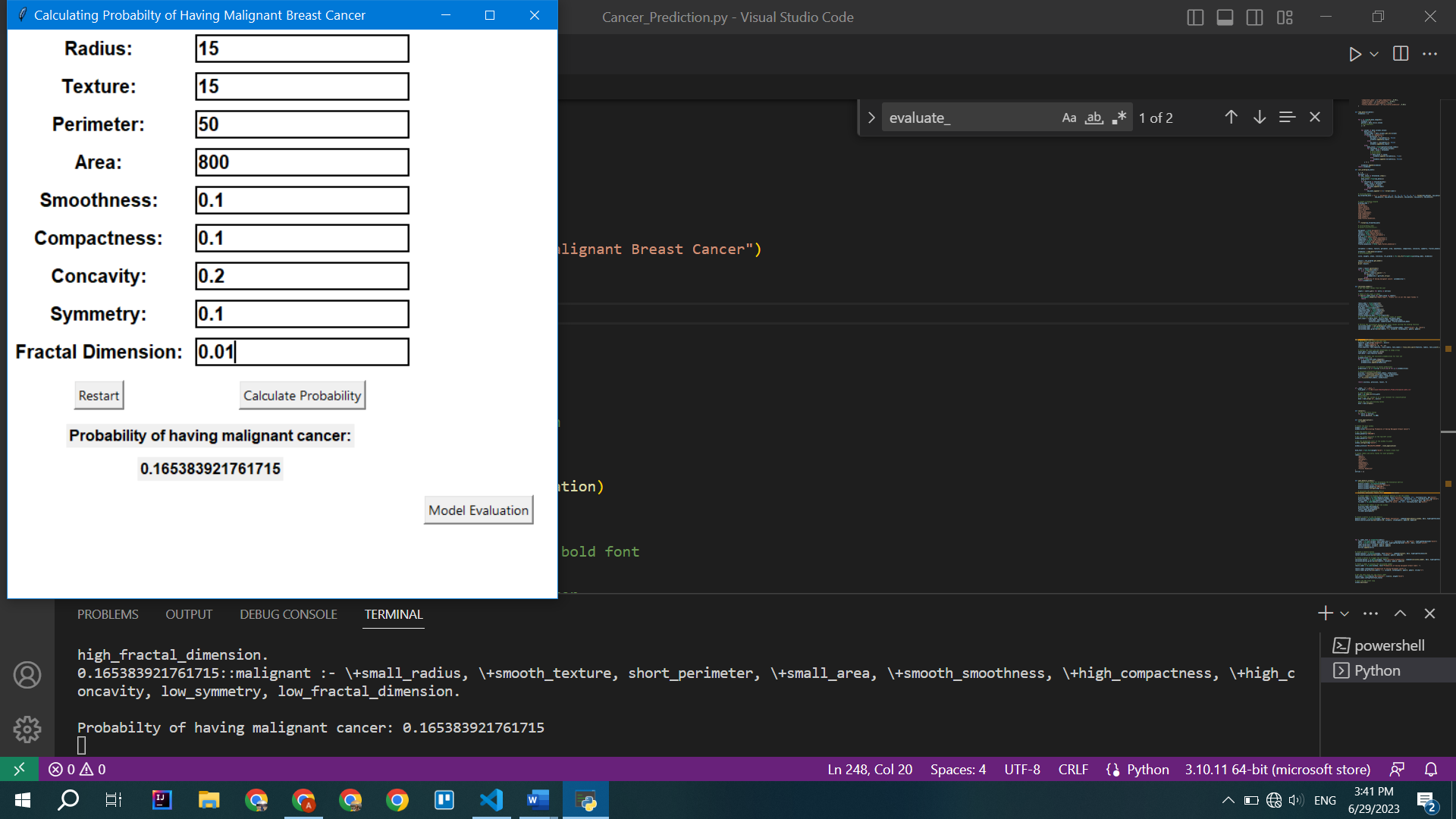
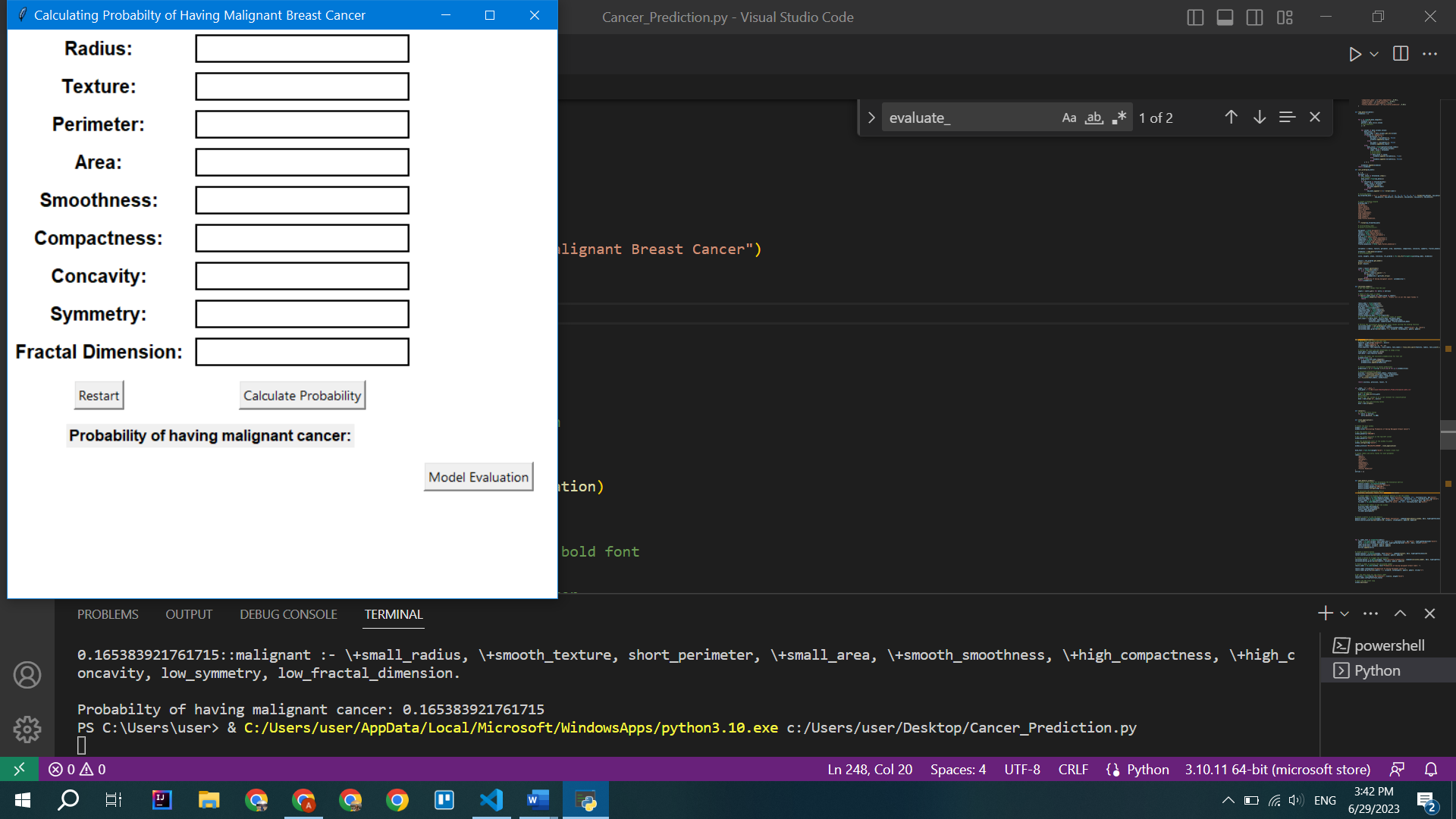
5.4 Defining functions: The code defines several functions for data processing and probabilistic reasoning. The function read\_data reads the data from the dataset and creates an array of evidence which will later be given to Problog. This function handles all the minor problems associated with reading the data from the dataset. For example, all the columns contain numeric values, except the ‘diagnosis’ column, so it is handled appropriately.

5.5. Calling Problog: The function 'call\_problog' takes the user input values, converts them into Problog format, and performs probabilistic reasoning using the Problog library. It calculates the probability of having malignant cancer based on the given input values and returns that probability. If any of the input values are empty, it shows an error message on the screen and asks to fill in all the input fields.

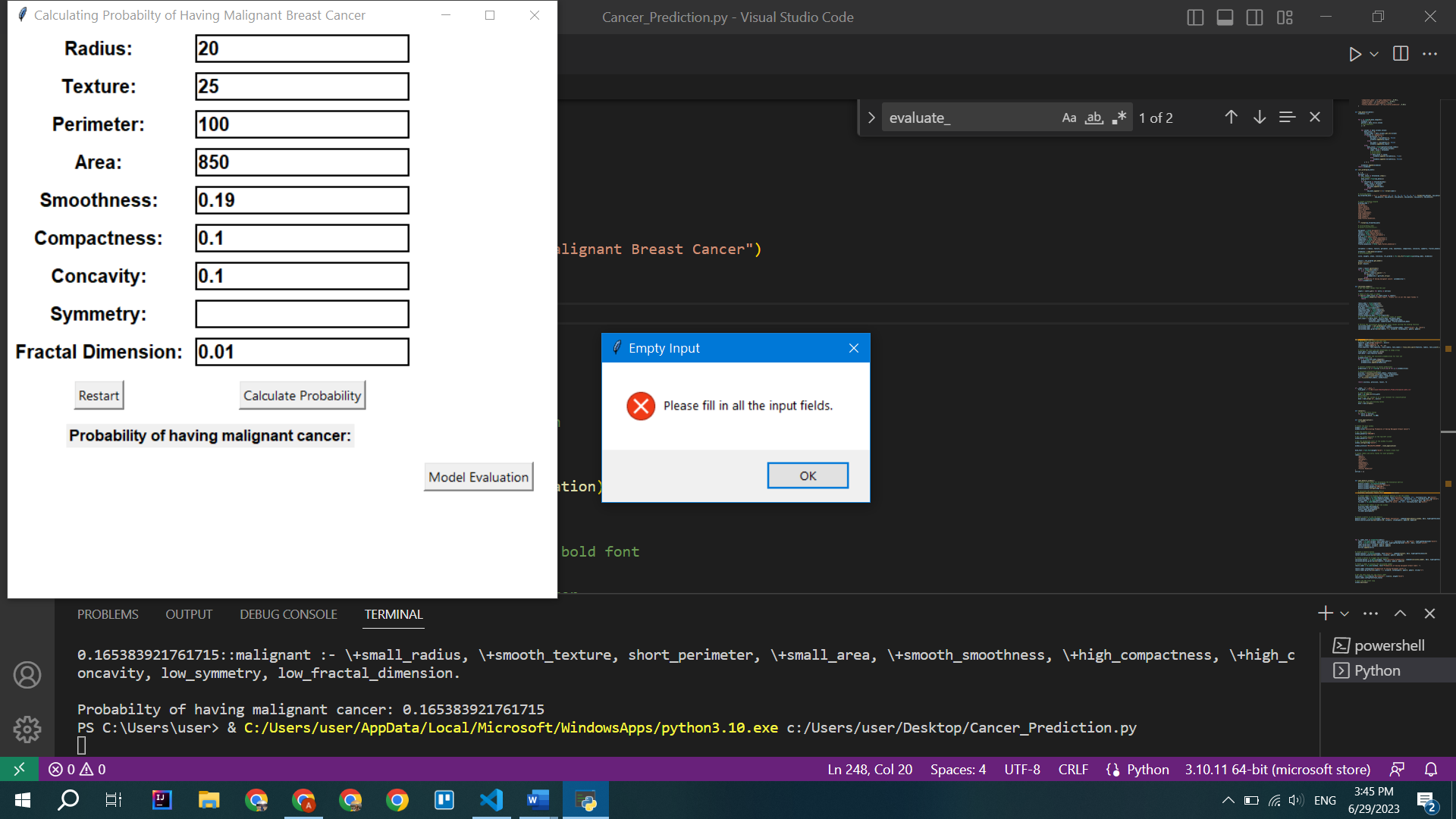
5.6. GUI creation: The code creates a graphical user interface (GUI) using Tkinter. It includes labels and entry fields for the input parameters. There are buttons to calculate the probability and restart the application. The calculated probability is displayed using a label. The restart button sets all the fields to their initial positions (i.e. empty) and allows to run the application again.

5.7. Event handling: The code handles events such as button clicks and retrieves the user input values. It calls the 'call\_problog' function to calculate the probability and updates the GUI accordingly.

5.8. Main program execution: The code executes the main program by loading the dataset, creating the GUI window, and starting the GUI event loop.

5.9. Below is the demonstration of the application:

First image is taken when the program is executed, and the second one is after the corresponding values are imputed and the probability is calculated. In this demonstration, all the parameter values imputed are below the threshold of malignant cancer, and we see that the resulting probability is 18%. Therefore, the accuracy of the algorithm is high.

If the user does not input one of the values they will receive an error message on the screen as below:  


For example, here I demonstrated the case when the user forgets to input the value for symmetry.

## **Evaluation Metrics**

We will evaluate the performance of a machine learning model using various evaluation metrics from scikit-learn library [4]. The model has been trained to classify instances into two categories: benign (B) and malignant (M). Malignant (M) is mapped to the value ‘1’ and Benign (B) is mapped to the value ‘0’. If the probability calculated by the program is higher than 50%, it is considered malignant, if it is less than 50%, the tumor is considered benign. We will discuss four evaluation metrics: accuracy, precision, recall, and F1 score. Additionally, we will provide an explanation of the code used for evaluation and the library employed in the process.

**Evaluation Metrics:**

***6.1 Accuracy:***

Accuracy measures the overall correctness of the model's predictions. It calculates the ratio of correctly classified instances to the total number of instances in the test set. Accuracy is useful when the classes in the dataset are balanced.

**The obtained accuracy for our model is 0.5614, indicating that it correctly predicts the class of approximately 56.14% of the instances.**

***6.2 Precision:***

Precision quantifies the model's ability to correctly predict positive instances among all instances predicted as positive. It is calculated as the ratio of true positives to the sum of true positives and false positives. Precision is particularly useful when the cost of false positives is high.

**In our evaluation, the precision of the model is 0.4340, suggesting that around 43.40% of the instances predicted as malignant are indeed malignant.**

***6.3 Recall:***

Recall, also known as sensitivity or true positive rate, measures the model's ability to identify positive instances correctly. It is calculated as the ratio of true positives to the sum of true positives and false negatives. A higher recall value indicates that the model is effective at capturing positive instances.

**In our evaluation, the recall of the model is 0.5349, indicating that approximately 53.49% of the actual malignant instances were correctly identified by the model.**

***6.4 F1 Score:***

The F1 score is the harmonic mean of precision and recall, providing a balanced measure of the model's performance. It combines both precision and recall into a single metric, giving equal weight to both measures. The F1 score considers both false positives and false negatives and is useful when there is an imbalance between the classes.

**The F1 score for our model is 0.4792, which suggests a reasonable balance between precision and recall.**

**Below is the summary of all the metrics:**

Accuracy: 0.5614

Precision: 0.4340

Recall: 0.5349

F1 Score: 0.4792

These metrics and their respective values help us understand the model's performance and guide further analysis and improvements if necessary.

**Code Explanation and Library Used:**

The code provided utilizes several libraries to evaluate the machine learning model's performance. Here is an explanation of the key aspects:

***Libraries Used:***

The code employs the scikit-learn library, a widely used machine learning library in Python that provides various tools for classification, regression, and model evaluation.

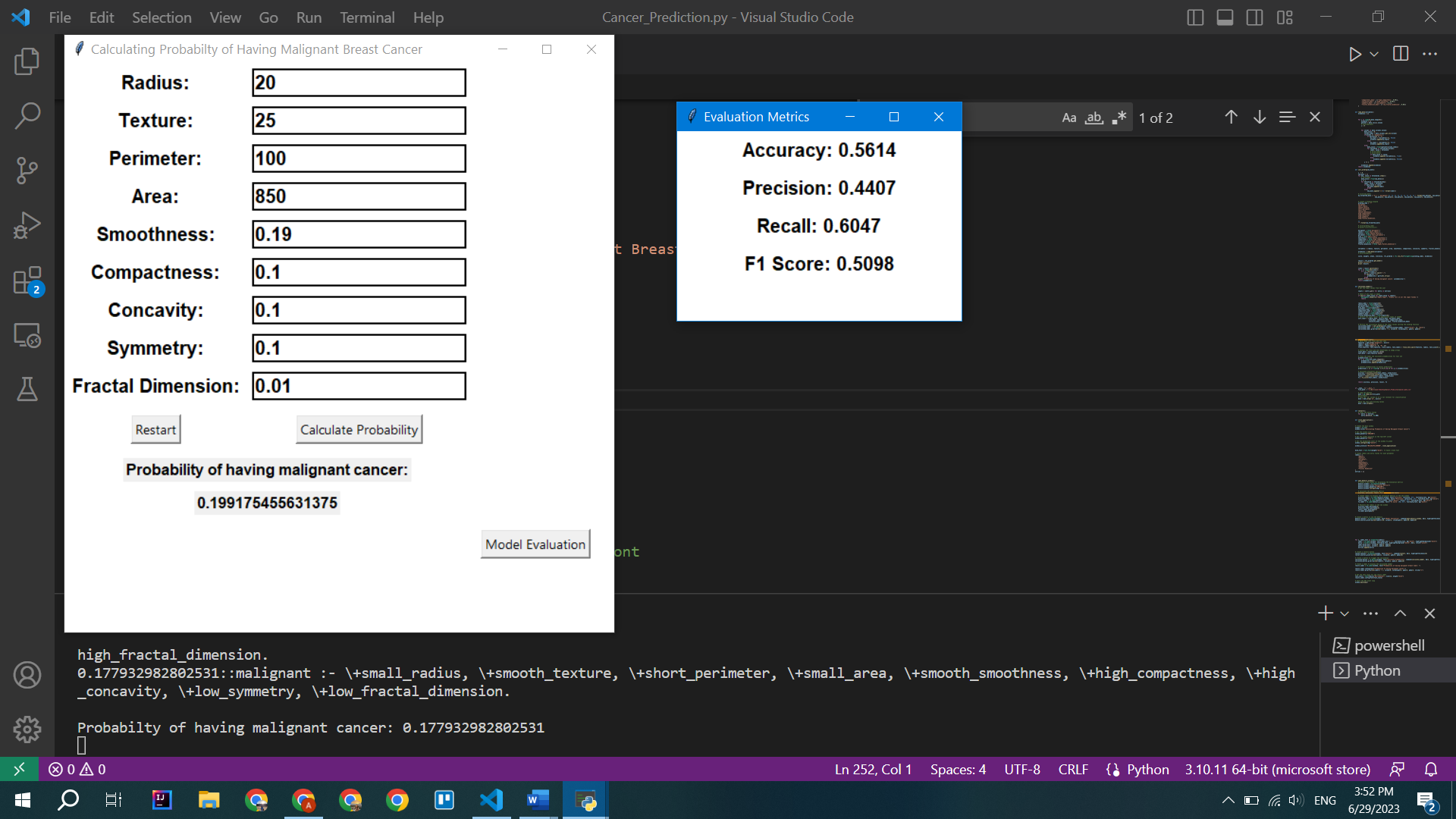
***Model Evaluation:***

The function evaluate\_model takes in the dataset and performs the evaluation of the model's performance. The dataset is split into training and test sets using the train\_test\_split function from scikit-learn. **The train set contains 80% of the data, and the train set contains 20%.** The model is trained on the training set and probabilities are calculated for the test set using the function call\_problog. The probabilities are then converted into binary predictions using a threshold of 0.5.

Finally, the accuracy, precision, recall, and F1 score are computed using the appropriate functions from scikit-learn (accuracy\_score, precision\_score, recall\_score, f1\_score).

***5.5 Demonstration of the evaluation metrics window***

We have the ‘Model Evaluation’ button, which, when pressed, shows all the evaluation matrics on a separate window as demonstrated below:

******

## **Conclusion:**

As a result, this project effectively applied a Problog-based strategy with a GUI interface for determining the malignancy of breast cancer. A Bayesian network is used by the created system to describe the connections between several breast tumor-related factors. The code report gives a thorough rundown of the implementation procedures, including library import, dataset pretreatment, Problog integration, GUI development, event handling, and program execution.

Accuracy, precision, recall, and F1 score were utilized as evaluation measures to evaluate the machine learning model's performance. The model achieved an F1 score of 0.4792, accuracy of 0.5614, precision of 0.4340, recall of 0.5349. These measures offer information about the model's accuracy, its capacity to correctly anticipate positive instances, and its harmony between recall and precision.

Overall, this study shows a successful method for predicting the malignancy of breast cancer using probabilistic reasoning and offers a user-friendly interface for entering parameters and generating probability estimates. The evaluation metrics found can serve as a reference for additional study and advancements.

## **References**

1. GUI Tkinter Library Documentation:

Python Software Foundation. "GUI Tkinter Library Documentation." Python 3.9.7 Documentation. Accessed September 2021. <https://docs.python.org/3/library/tk.html>

2. Pandas Library Documentation:

Pandas Development Team. "Pandas Library Documentation." Pandas 1.3.0 Documentation. Accessed September 2021. <https://pandas.pydata.org/docs/>

3. Problog Library Documentation:

Problog Development Team. "Problog Library Documentation." Problog Latest Documentation. Accessed September 2021. <https://problog.readthedocs.io/en/latest/>

4. Sklearn Library Documentation:

<https://scikit-learn.org/stable/index.html>

5. Problog Library in Python Usage Documentation and Examples:

Problog Development Team. "Problog Library in Python Usage Documentation and Examples." Problog Latest Documentation. Accessed September 2021. <https://problog.readthedocs.io/en/latest/python.html>

6. Reading and using a dataset in Problog:

AGH University of Science and Technology. "Reading and using a dataset in Problog." Artificial Intelligence Group - AGH University of Science and Technology. Accessed September 2021. <https://ai.ia.agh.edu.pl/en:dydaktyka:problog:lab1>

7. LFI model Documentation:

Problog Development Team. "LFI model Documentation." Problog Latest Documentation. Accessed September 2021. <https://problog.readthedocs.io/en/latest/modeling_basic.html>

8. M. Zaman and C. -H. Lung, "Evaluation of machine learning techniques for network intrusion detection," NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium, Taipei, Taiwan, 2018, pp. 1-5, doi: 10.1109/NOMS.2018.8406212.