Table of Contents

[Introduction 2](#_Toc134741279)

[Methodology 3](#_Toc134741280)

[**Stage 1: Algorithm design** 4](#_Toc134741281)

[Dataset description 5](#_Toc134741282)

[Exploratory data analysis 5](#_Toc134741283)

[Data Visualisation 9](#_Toc134741284)

[Predictive Analysis 12](#_Toc134741285)

[Feature Scaling 12](#_Toc134741286)

[Train Test Split 14](#_Toc134741287)

[Model comparison 14](#_Toc134741288)

[**Stage 2: Algorithm Implementation** 17](#_Toc134741289)

[Hyperparameter tuning. 17](#_Toc134741290)

[Trained model and result 18](#_Toc134741291)

[Save Model 19](#_Toc134741292)

[Model Interpretation 20](#_Toc134741293)

[Export artefacts for WebApp 21](#_Toc134741294)

[**Stage 3: Software Deployment** 23](#_Toc134741295)

[Import of Web App 23](#_Toc134741296)

[Load artefacts 23](#_Toc134741297)

[Web App Layout 24](#_Toc134741298)

[Data showcase tab 25](#_Toc134741299)

[Cell estimator Tab - ESTIMATION FUNCTION 25](#_Toc134741300)

[Web App Sample showcase 27](#_Toc134741301)

[Evidence of Testing 32](#_Toc134741302)

[Reflection 32](#_Toc134741303)

# Introduction

The purpose of this report is to demonstrate real world applicability of machine learning within the scope defined in ST1G Capstone project requirements. In this report, the Cancer Data from Kaggle [1] is used for data analysis and the training of machine leaner.

Cancer is one of the most fatal and incurable diseases in the world, despite being discovered as early as in the 19th Century, they are still as deadly as ever. Most of the patients survived due to an early discovery of the symptom, which makes an early diagnosis of the disease crucial to the patient's survival. As patient with early diagnosis are able to treat the disease earlier and increase the chance of recovery. Furthermore, an accurate classification of the tumours, such as benign tumour, can prevent the patient from undergoing treatment that are unnecessary. Since the treatment itself can be detrimental to the patients as well, such as chemotherapy. Therefore, correct diagnosis of the disease led by accurate classification of malignant and benign cells will be an interesting topic to investigate and the result of the research will contribute to the field of medicine.

Machine Learning, as an artificial intelligence technique, has advantage in classification by nature. The technique will be suitable for providing a solution to this problem, by classifying tumour cells based on their critical features, such as radius, concave point, smoothness and more.

The rest of the report well demonstrate the implementation of a machine learning model and a web application. The dataset [1] will be imported using Python in Visual Studio Code IDE. Followed by exploratory data analysis and data visualisation. Issues found in Exploratory data analysis will be solved by data pre-processing. And lastly, predictive analysis and web application will be implemented using python machine learning and web development libraries and packages such as Scikit-Learn [2] and Streamlit [3].

# Methodology

In this project, the following step of implementation is followed:

1. Select dataset that contains cancer cell geometric data.
2. Import necessary data analysis/manipulation modules and machine learning modules for later steps.
3. Perform exploratory data analysis, in search of data integrity, data type, number of samples, number of features and more. Determine linearity of the data.
4. Select machine learners that are suitable for classification problem. And perform preliminary predictive data analysis to choose the best candidates amongst the algorithms.
5. Hyperparameter Tuning, tune the hyper parameter of the selected best performing algorithm to improve its performance furthermore.
6. Evaluate the model’s performance using classification report.
7. Save the trained, hyper parameter tuned model for later use.
8. Perform model interpretation analysis using Lime [4] package, observe model behaviour.
9. Export the model and data visualisation artefacts (figure, classification report, object) to Streamlit web application.
10. Implement a cancer cell prediction web application prototype.

# **Stage 1: Algorithm design**

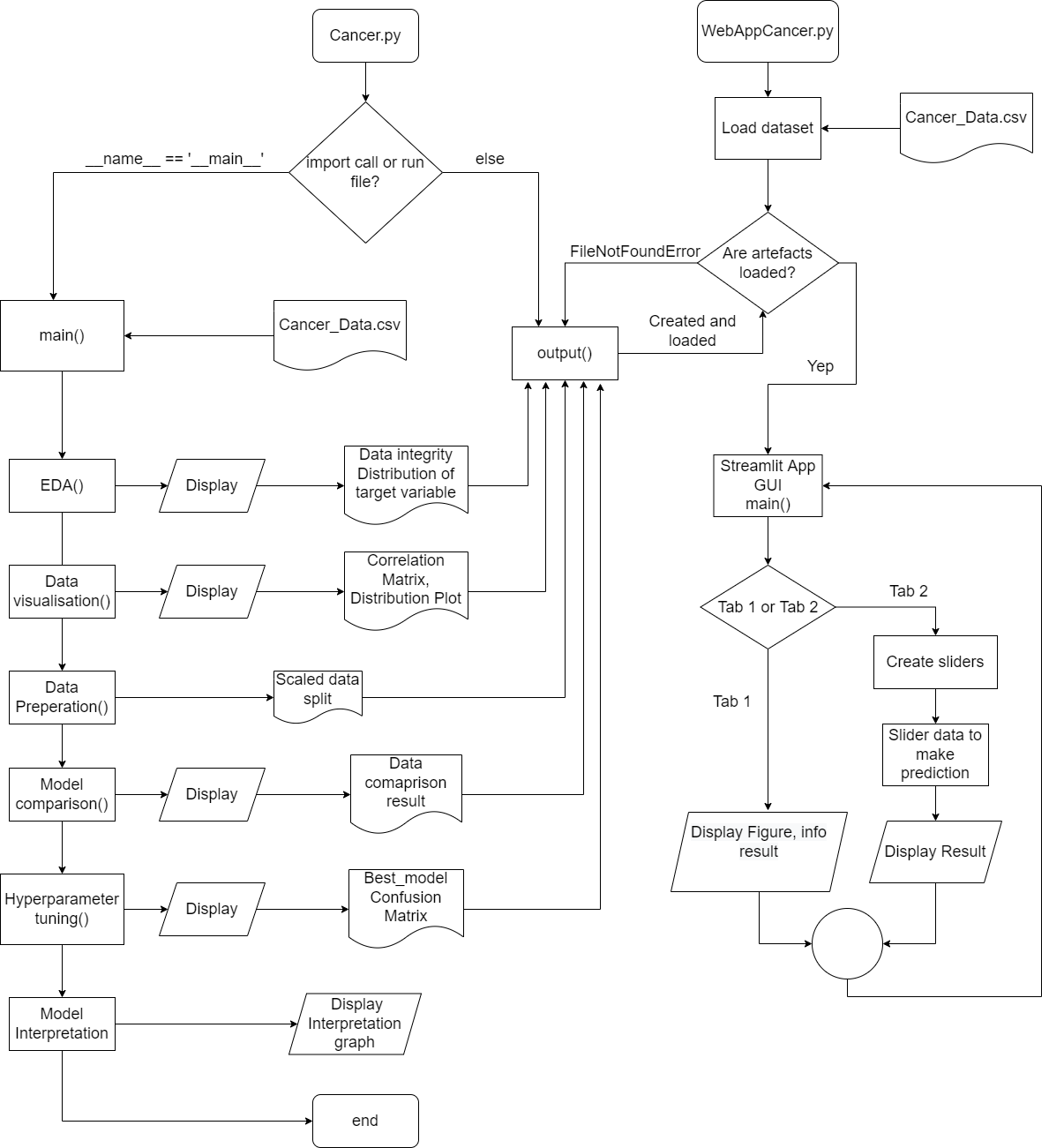


Figure Flow Chart of algorithm design

Figure 1 illustrates the program’s design. There will be two python modules built for this project. The first one is Cancer.py. This module will be responsible for the Exploratory Data Analysis, Data Visualisation, Data pre-processing, predictive analysis (Model training), hyperparameter tuning, and model interpretation. The artefacts such as figure, trained model, classification report generated by this module will be captured and stored in local directory by the output function. WebAppCancer.py will import artefacts generated by Cancer.py for showcase and interactive cell estimation using the trained model.

## Dataset description

The dataset used in this project is Cancer Data from Kaggle repository [1]. It is available publicly and similar dataset can be found in scikit-learn [2] machine learning package, namely Breast Cancer Dataset. However, in this project, the CSV file from Kaggle is used. The dataset has 569 samples of tumour cell’s geometric data generated from image of tumour cells. It has 30 features, and 1 classification attribute. The feature consists of data like the mean, standard error, worst/largest value of the radius of cell, compactness of cell, perimeter of cell and more. The target attribute is separated into two categories. With B meaning the sample cell is benign, and M meaning that the sample cell is malignant. Therefore, in order to predict the tumour cell, these data will be fed to the machine learner. But first, I need to perform exploratory data analysis to examine the data.

### Exploratory data analysis

In this analysis, a virtual environment is built using Microsoft Visual Studio Code IDE. Python Version 3.11.3 is used. And the following libraries and packages are imported into the virtual environment.

Text

Description automatically generated

Figure modules imported in Cancer.py.

As shown in Figure 2, number of modules is imported. Scikit-learn are imported for predictive analysis, and data pre-processing. Matplotlib and Seaborn are imported for data visualisation. Pandas and NumPy are imported for Data Exploration. Lime is imported for model interpretation. And lastly, Joblib is imported for models and artefact saving.

Text

Description automatically generated

Figure Import dataset.

Figure 3 shows the dataset import.

The following are the question that I would like to answer. Some of the snip shots codes are compiled from the jupyter notebook, nevertheless the program is **not** made in jupyter notebook format (.ipynb). So, rest assured.

1. What does the dataset look like?

Text

Description automatically generated

Figure First Five row of the dataset.

1. How many rows and column does the dataset have?

Text

Description automatically generated

Figure Shape of Dataset

1. What is the data type of these data, and what are the features?



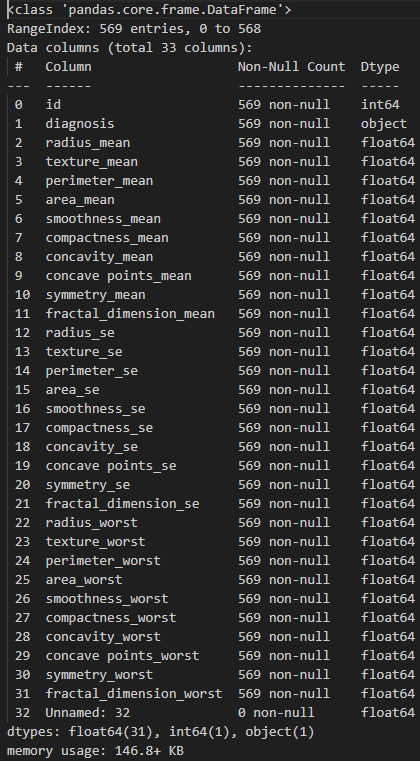


Figure Data type of dataset.

So, we can tell from Figure 6 that there are three types of data. They are int, object, and float.

1. How many benign cases and malignant samples are in the dataset?

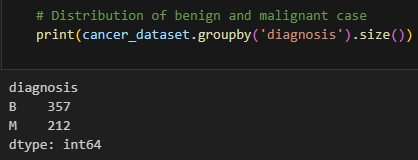


Figure Number of Benign and Malignant cases

From Figure 7, we can see that the dataset is slightly imbalanced with more Benign cases. Nevertheless, the imbalanced will have minimum impact on the model’s performance.

1. Is there any missing value?



Text

Description automatically generated

Figure Check Missing Value

From Figure 8, we can tell that none of the features have value missing. However, there are a feature, namely Unnamed: 32, have 569 missing values.

Based on the exploratory data analysis, there are a few things that draws attention.

1. From Figure 4, we can tell that the data type of the target attribute is object, which will make the later analysis difficult.
2. From Figure 4, we can tell that ID feature has no correlation with the samples target attribute, the tumour cell’s state is not determined by its assigned ID. Hence we will need to remove it in the next stage of analysis.
3. From Figure 4, 6, and 8, a feature with 569 missing values, denoted as NaN is shown in the analysis. The result indicates that this is an empty column in the CSV file, which will need to be remove as well. Because some machine leaner such as the K nearest Neighbour model cannot take empty data as input.

## Data Visualisation

To visualise the data, we first need to remove unwanted feature and column as described above. To do that, we will use these two algorithms:



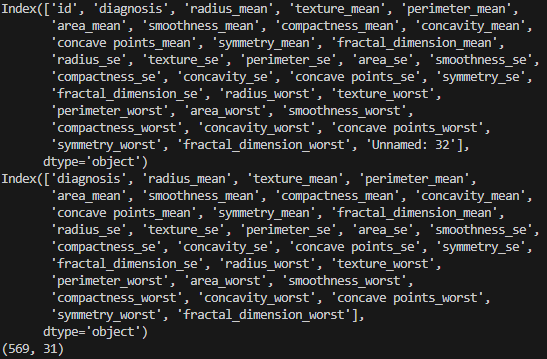


Figure Remove before and after comparison.

In Figure 9, we can see that the two columns are removed as intended, and other features remain intact.

With that out of the way, the data visualisation can proceed.

Also, the diagnosis variable needs to be changed from B, M to 0, 1 in order to plot figures.



Figure Substitute object variable to binary

**Heat map – Correlation Matrix**

**Text

Description automatically generated**

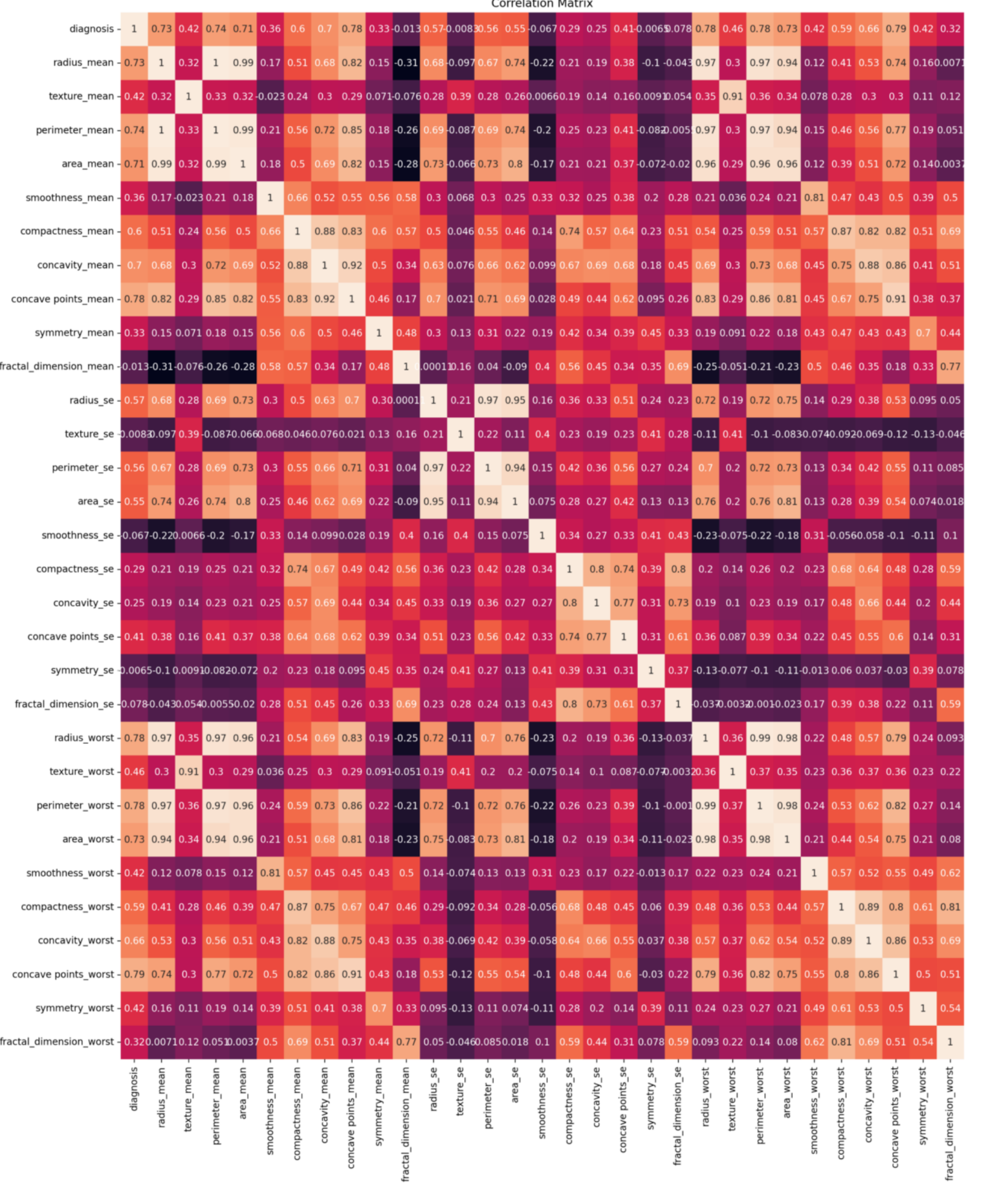
****

Figure Heatmap of Correlation Matrix

**Distribution of features and target**

Text

Description automatically generated

**Calendar

Description automatically generated**

Figure Distribution plot

The distribution plot illustrates the distribution of features and target. From the plot, we can tell by the overlapping of the graph, that we are dealing with non-linearly solvable data. Hence, in terms of algorithm choice, I posit that that machine learners that is suitable for non-linear issue will perform better.

## Predictive Analysis

To use the data to train the machine learning model, several procedures need to be undertaken.

### Feature Scaling

As shown in Figure 4, we can tell that the magnitude between data is drastically different. Some varies in decimal, and some varies in ten’s and hundred’s due to the difference in unit . Some machine leaners are inherently sensitive to the magnitude and scale of the data. For example. Distance-based machine leaner such as Support Vector Machine (SVM) and K Nearest Neighbour (KNN). They use the distance between each data points to estimate their probability [5]. Therefore, scale the data into lower or similar magnitudes will help the algorithm to perform better.

Example:

Text

Description automatically generated

Figure Unscaled

We can see that the accuracy is 94.7% using unscaled data.

Text

Description automatically generated

Figure Scaled.

The Accuracy of this SVM algorithm improve from 94.7% to 98.2%.

Therefore, after splitting the feature columns and the target column:

Text

Description automatically generated

Figure Feature scaling using standard scaler.

Text

Description automatically generated

Figure Feature Columns after scaling.

We will use StandardScaler() from Scikit-Learn to standardise our feature columns X. The value after the feature is scaled represents the standard deviation the value is away from the mean value of the feature. Calculated by this formula:

## Train Test Split

After scaling our data, a split of the data into training set and validation (Test) set is performed using trian\_test\_split().

Graphical user interface, text

Description automatically generated

Figure Train Test Split

With the training set vs. Testing set ration to be 80:20. And a fixed random state so that the performance result will not vary in order to have a more stable comparison in the next stage.

## Model comparison

In order to choose the best performing model for this task, an algorithm comparison is performed using K-fold Cross Validation Technique.

Four candidates are chosen based on there competence in solving nonlinear classification problems.

* Support Vector Machine
* Classification and Regression Tree (CART), also known as Decision Tree
* K Nearest Neighbour
* Naïve Bayes

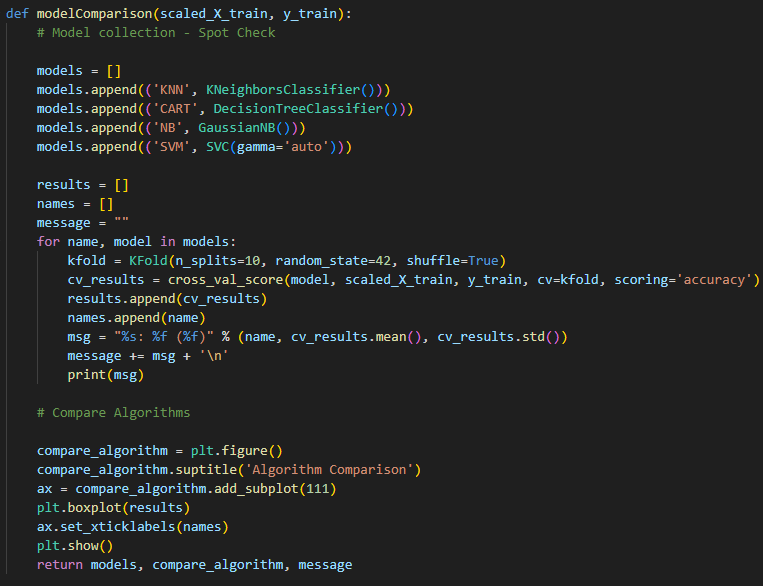


Figure Model Comparison Code - K fold Cross Validation

In this algorithm, the train-test-dataset is split 10 times (n\_splits = 10) covering all parts of the train-test-dataset. Using this technique can help us to avoid bias results caused by over-fitting the model, which means the model is performing excellently in the training process but perform poorly when it meets unseen data. The result of the validation is shown below.

Chart, box and whisker chart

Description automatically generated

Figure Comparison result

From the result, we can see that all the candidates perform well, there accuracy score all reaches over 90%. The phenomenon can be explained by the fact that the dataset is relatively small. With only 569 sample, the result may not be indicative. Nevertheless, we can see that SVM outperform the rest of the candidates with the highest accuracy score of 98%. From the box plot, we can tell that it performed stably with a few outliers lying at approximately 93%, which is still a very good performance. And the standard deviation supports the fact that it is performing stably as well.

Hence, for the choice of machine learner, support vector machine is the model that will be deployed in the Web Application for cell estimation.

# **Stage 2: Algorithm Implementation**

### Hyperparameter tuning.

With the best model for solving this problem identified, model training will be in process. But first, we can tune the Hyperparameter of the model to increase it performance on the model using GridSearchCV().

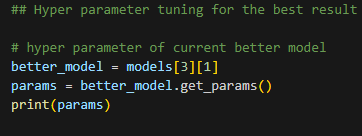


Figure Current Param Code

**{'C': 1.0, 'break\_ties': False, 'cache\_size': 200, 'class\_weight': None, 'coef0': 0.0, 'decision\_function\_shape': 'ovr', 'degree': 3, 'gamma': 'auto', 'kernel': 'rbf', 'max\_iter': -1, 'probability': False, 'random\_state': None, 'shrinking': True, 'tol': 0.001, 'verbose': False}**

In these parameters, four of them are worth mentioning.

Support vector machine works as a separator, for example, when the datapoint (support vector) is plotted in a 3D space, support vector machine tries to find a hyperplane that can separate data in to N categories. Thus:

1. C – Fault tolerance, how many “mistakes” is it allowed to make, with higher C meaning more and lower meaning less. Sometimes, the lowest is not always the best.
2. Gamma – How concave can the hyperplane be, with more concaveness/curvature enabled (smaller gamma), the hyperplane will be more “uneven” and fit the support vectors more closely, and vice versa. Again, smaller value is not always the best.
3. Kernel – for support vector machine, there are a few kernels that the SVM can use for calculation, mostly RBF and Linear. Since we are dealing with a nonlinear classification problem. We will use the RBF kernel.
4. Probability – SVM is not a probability-based machine leaner (unlike Naïve Bayes). However, probability can be enabled to mimic a probabilistic estimation. For the Web Application’s function, the probability will be set as TRUE.



Figure GridSearchCV code

20 folds of calculation using the above configurations are conducted, a total of 840 attempts. The function then returns the best configuration, which is:

{C =10, gamma = 0.01}

### Trained model and result

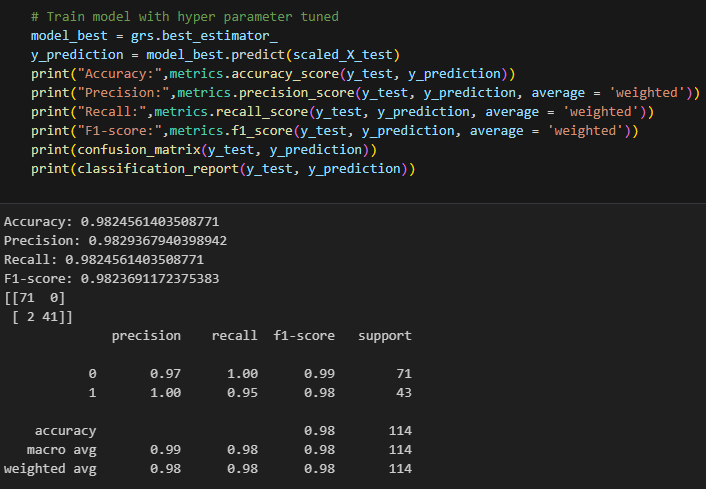


Figure GridSerachCV Result

The result showed a small improvement.

Chart

Description automatically generated

Figure Confusion Matrix of the model

### Save Model

And finally, save the trained model for Web Application.



Figure Save model code.



Figure Saved model example.

### Model Interpretation

A train model does not give out any information about what the reason that the model is making the prediction. Thus, the Lime package [4] is used for model interpretation.

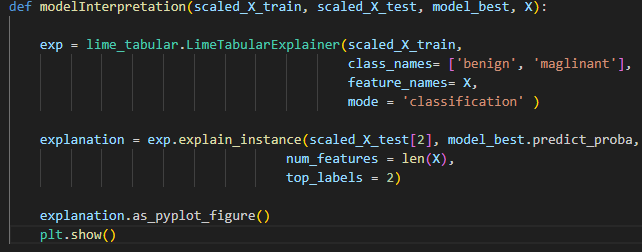


Figure Model interpretation using Lime.

Chart, treemap chart

Description automatically generated

Figure Interpretation Figure

From the above figures, we can see that the scaled\_X\_test[2] is chosen to make a prediction. The above interpretation result indicates that the model is 92% confident that the result is a benign case. And the vertical bar chart illustrates the factors that influence the model to make the prediction.

### Export artefacts for WebApp

After the model is trained, it is necessary to program a function that export all the relevant artefacts (figure, model, object), so that the web application can utilise it without having to rerun Cancer.py every instance when user is using the application.

Graphical user interface, text

Description automatically generated with medium confidence

Text

Description automatically generated

Figure Main function

Text

Description automatically generated

Figure Output function

As shown above, the output function works similarly to the main function, with difference being that the output function will use the Joblib package to export and save the artifacts as binary save files into the same directory Cancer.py is in.

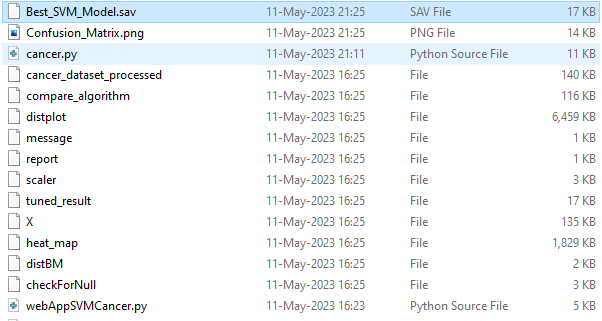
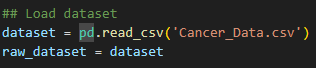


Figure Artefact saved.

# **Stage 3: Software Deployment**

## Import of Web App



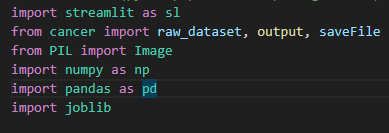


Figure Web App Import

For the Web application, Streamlit is used to create a web interface. Besides the aforementioned module import used in Cancer.py, PIL which stands for python Pillow built in imaging library is also used to display Confusion\_Matrix.fig.

## Load artefacts

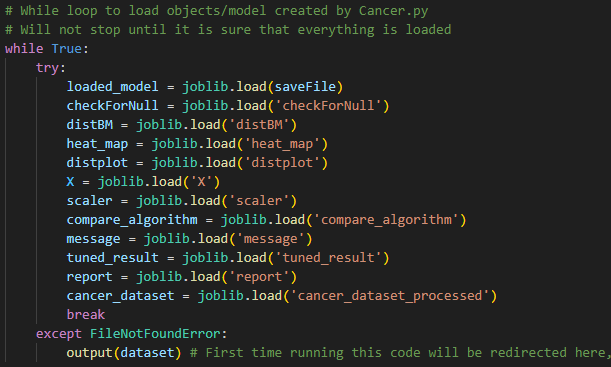


Figure While loop load artefact.

For first time users, there will not be anything to load. So, a FileNotFoundError will raise, redirecting the program to the output function imported from Cancer.py, thus generating all the artefact needed.

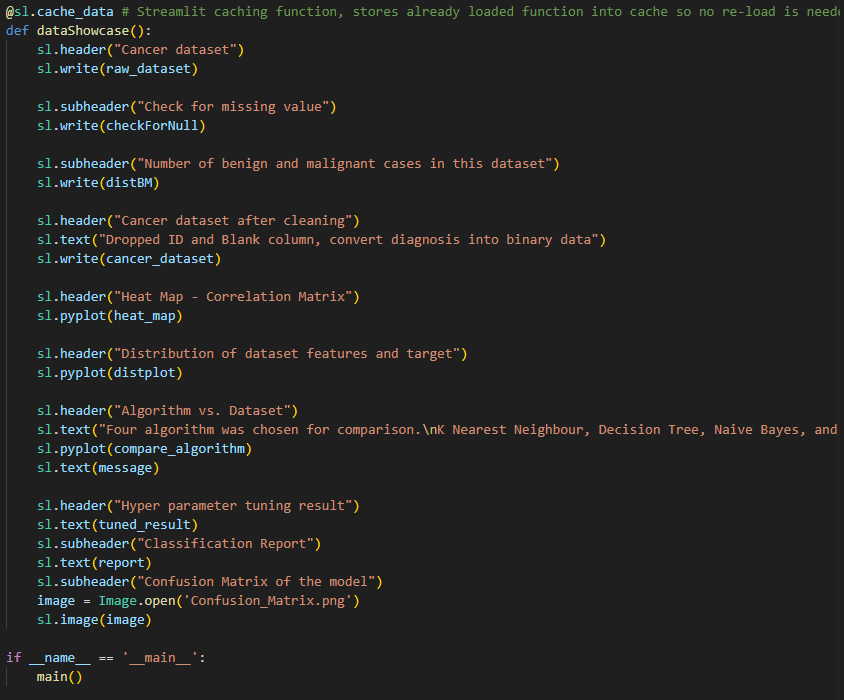
## Web App Layout

Text

Description automatically generated

The web application is contained in the main function. It has two tabs, Data Showcase and Cell estimator. Which does what their name suggested.

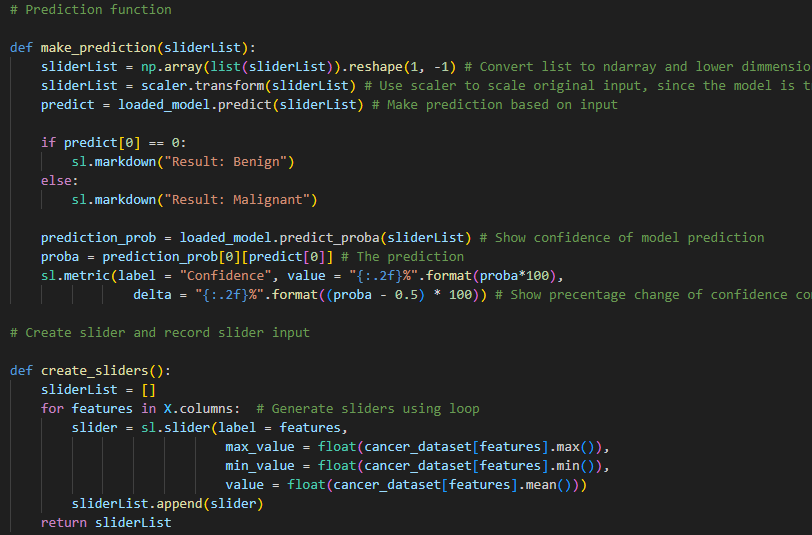
### Data showcase tab



All the results and figured generated from Cancer.py is placed in this tab. It is worth mentioning that the Streamlit function @sl.cache\_data is used. Due to the nature of Streamlit, the webpage will refresh itself each time the user made an input. In this case, when the slider moves, the whole application will refresh itself. The cache function solved this problem by saving the unchanged data and skip the execution of the function that has already been executed. The speed of estimation vastly improved.

### Cell estimator Tab - ESTIMATION FUNCTION

This is the core or the project. It consists of slider created using the feature name of the dataset using for loop. And the allowed minimum and maximum input is set to the minimum and maximum of each feature. The default input is the mean of the features. The result will be shown on this tab providing the estimated result (Benign, Malignant) and the Confidence of the model prediction.



## Web App Sample showcase



Calendar

Description automatically generated with medium confidence





A screenshot of a computer

Description automatically generated with medium confidence

# Evidence of Testing

**Screen shots of testing has been provided in the previous section of the report.**

Test plan**:**

In terms of Test plans. The implementation of machine learner training was done in Jupyter Lab using Jupyter notebook. Each step of the implementation was run individually to ensure the data structure and intended input and output is as expected. It is further tested when the program is modularised into functions. There are resemblances between the test plan adopted in this project and unit testing.

As for the software deployment stage. Initially, the webpage content and function are added block by block. Upon completion, the development enters a lite version of user acceptance testing. Tested by other individual (my partner) besides the developer (me). A suggestion/complaint was received from the tester, which is the program ran too slow. It is because the output function was not created at that time of development. The lack of output function resulted in the web application execute the Cancer.py program every time the user moves the slider (input a value). Nevertheless, the inefficiency of the code is fixed.

# Reflection

In this project, I experienced most of the part of the software development life cycle, including planning/designing, analysis on data and possible implementation, implementation of both the model training module and web application, and the testing of the program and its integration. Throughout the development, I faced many hurdles. Since the beginning of the project, setting up virtual environments had spent a considerable amount of time. I have struggled for approximately an hour on this matter. And the situation is resolved in minutes by researching online. It demonstrates that in the field of software development, seeking help online as needed is of utmost importance. Furthermore, understanding the data structure and data type of each parameter will help a developer to quickly debug in the coding process. In the webapp, the slider input is stored as a list. Not only that converting it to a ndarray is important, but also the dimensionality needs to be reduced for support vector machine to make prediction. Which raises another point, that is, documentation is a programmer’s best friend. Approximately 60% of the bug I had encountered is solved by studying the documentation.

# References

|  |  |
| --- | --- |
| [1] | E. Taha, “Cancer Data,” 2023. [Online]. Available: https://www.kaggle.com/datasets/erdemtaha/cancer-data. [Accessed 25 April 2023]. |
| [2] | Scikit-Learn, “Scikit-Learn Machine Learning in Python,” March 2023. [Online]. Available: https://scikit-learn.org/stable/. [Accessed 25 April 2023]. |
| [3] | Streamlit, “A faster way to build and share data apps,” 2023. [Online]. Available: https://streamlit.io/. [Accessed 1 May 2023]. |
| [4] | P. Banerjee, “Explain your model predictions with LIME,” 2020. [Online]. Available: https://www.kaggle.com/code/prashant111/explain-your-model-predictions-with-lime. [Accessed 1 May 2023]. |
| [5] | Analytics Vidhya, “Feature Engineering: Scaling, Normalization, and Standardization (Updated 2023),” 11 April 2023. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/#What\_Is\_Normalization?. [Accessed 3 May 2023]. |