Analysis Report Template

Student Full Name	Hamza Hassan	Student ID No.	W2044381

Case Study Title

Case Study (A): Predicting Cancer Patients Mortality Status.

Research Question: Does machine learning have the potential to assist doctors in predicting those who will survive breast cancer or not?

[43 Marks]

Task (1) – Domain Understanding: Classification

The doctors decided that classification modelling is required. **Indicate in the table below** for each of the listed **variables in your data which ones you should RETAIN** and can be included in the classification modelling of Breast Cancer Mortality (Alive vs. Dead) and **the variables you should DROP (REMOVE). Justify your decision logically and/or by research** (include in-text citation)

[3 Marks]

Variable Name	RETAIN or DRO	Brief justification for retention or dropping with in-text citation.
Patient ID	Dropped	The reason I thought to drop this variable was since it is a Unique Identifier for all patients, it cannot bring anything useful to the model. It contains nothing but meaningless information, has nothing to do with the Patients' Health Condition or even the patients Survival. So, through this I just though it wouldn't bring any use in our prediction of Breast Cancer Mortality and prediction the Patients Survival Status.
Month of Birth (In my dataset renamed to Birth Month)	Dropped	Month of Birth was also dropped because of the fact of how unrelated it is to Survival Status, there is no direct link of Birth Month to the Mortality of a Breast Cancer Patient, it doesn't inherently affect health outcomes or lifespans. So, I've chosen to drop this aspect from the model.
Gender	Dropped	I dropped Gender because I realised the Breast Cancer Dataset is majority Female, which is not ideal. It could lead to a bias and unreliable prediction For males in the dataset. The model needs stronger variables than Gender In my opinion to base its predictions on that's why I choose to drop it, even Though through research I heard that woman is more likely to have Breast cancer but just based on simplicity and more accurate predictions I chose to leave it out.
Age	Retained	I thought Age should be kept as it could be a key factor in predicting health Outcomes. It can have a high influence on how the body may respond to treatm And could affect livelihood of survival. So, in my opinion I think this variable Highly collates with mortality.
Occupation (In my dataset I Renamed as	Dropped	Dropping Occupation was a must, it doesn't collate to a patient's health or canc Prognosis. It's a more general attribute that is unrelated to the specific health Outcomes I'm trying to predict. Adding this would overcomplicate the model and Reduce its accuracy in prediction, so I think it's better I leave this variable out.

Profession)		
T Stage	Dropped	I dropped the T Stage column because it mainly shows, Tumor Load which is already covered better by 6th Stage and Grade. Keeping all three felt repetitive, and I didn't want to confuse the model with overlapping features. I decided to keep Grade because it shows how aggressive the cancer is, and 6th Stage gives a broader view of the cancer's stage, so they felt more useful overall.
N Stage	Retained	This variable I also though should be kept and retained in the model, as it Describes whether the Cancer has spread to the nearby lymph nodes. The Lym Node involvement is a strong indicator of Cancer Progression and can Significantly affect Survival Rates. So evidently, through research I discovered t The N Stage should be kept, and with the T stage variable would be another Key aspect in capturing a Clear Prediction of Survival with these Breast Cancer
6th Stage	Retained	The 6th Stage variable refers to the overall cancer staging, which combines both T-stage and N-Stage to provide a more complete picture of the Cancer's Spread. The Stage in which the Cancer is probably the most Reliable predictors of survival, as it can show how far the Cancer has spread Already. So, this variable would be key to keep in the model the enhance the prediction further.
Differentiated	Dropped	Dropping the variable Differentiated was needed because it seemed to be Very similar to the information which was already provided in Grade which i Kept. Grade also tells us how abnormal the cancer cells are, so keeping Both Differentiated and Grade would in my opinion confuse the models I'll be Using down the line like Logistic Regression which is sensitive to redundant Features. I decided between both variables that Grade is much more Clearer and more consistent to keep, so I dropped Differentiated for simplicity And a better model performance.
Grade	Retained	Keeping Grade in the model, I thought would be important as it describes How abnormal the Cancer Cells are under a microscope. This is another key Aspect of detecting Cancer Status in Patients that is done in hospitals, So I thought I would be wrong to remove this variable.
A Stage	Dropped	I thought to drop A Stage just based on the fact it brings no meaningful Information beyond what the other variables like Grade can bring, and it may be Redundant. I would rather simplify the Model with needed variables rather than adding not needed variables and overcomplicate the Model.
Tumour Size	Retained	This is one of the most needed variables for our predictions to be as Accurate as possible. It's a key variable in determining how advanced the

		Cancer is in size. It's a straightforward and vital feature in predicting Survival.
Estrogen Status (Written as Estrogen Levels in Dataset)	Retained	Estrogen Status (Levels) are essential and must be kept in my opinion, just Since Estrogen Receptors (ER) status can tell us whether the Cancer cells are Estrogen. This would help a lot more if this variable is obtained, it could determine survival predictions, making it essential for our model.
Progesterone Status (Written as Progesterone Levels In Dataset)	Retained	Like Estrogen Status, Progesterone Status can affect Cancer Growth, so Including this variable in the model I believe would further enhance the Models' ability to predict survival as would Estrogen Status.
Regional Node Examined	Dropped	This Variable indicates how many lymph nodes were examined to assess the Spread of Cancer. But I chose to drop this variable because it doesn't refer To how many were positive lymph nodes that were tested just the number of Lymph nodes, it doesn't show the severity of the cancer and wouldn't help Predict Mortality Status or Survival Months in my opinion. I would rather the Model focuses on variables that bring meaningful information and avoid using This variable to help with predictions so I decided to drop.
Regional Node Positive	Dropped	Regional Node Positive refers to whether the Cancer has spread to the Regional Lymph Nodes. The variable again is strongly related to N Stage, but What I realised is that N stage captures the lymph node involvement in cancer Much cleaner than Regional Node Positive. So, I think keeping this Variable would just cause redundancy, which would worsen the modelling like Logistic Regression. Again, for simplicity I chose to drop Regional Node Positive and keep N Stage instead.
Survival Months (Name as Progression Free Months in My dataset)	Retained	Survival Months is the length of time a patient has survived since being Diagnosed. It directly links and reflects the patient's outcome and is essential For our model in predicting the number of months a patient could survive. Retaining it provided the target variable for our model, and would be a key Part in boosting the accuracy of it.
Mortality Status (Named as Survival Status In dataset)	Retained	This variable indicated whether a patient is alive or deceased, and it's the most Direct indicator of the outcome I'm trying to predict. This is one of the most Crucial Variables in evaluating the success of the model and for making Survival predictions, So I believe 100% it should be retained.
References		

https://www.who.int/news-room/fact-sheets/detail/breast-cancer#:~:text=Female%20gender%20is%20the%20strongest,of%20management%20as%20for%20women.

https://www.cancerresearchuk.org/health-professional/cancer-statistics/statistics-by-cancer-type/breast-cancer/incidence-invasive

MODULE: (2024) 5DATA002W.2 Machine Learning and Data Mining
Coursework Full Briefing & Specifications Document (PDF)

Task (2) – Exploring and Understanding Your Dataset

With the aid of your Final Python Notebook 1, for your RETAINED input variables and your class "Target" variable, produce 1)**basic descriptive stat**s and 2)**variable scale type, then** 3)p**lot the distribution of your target variable**. (Paste the three screenshots of code OUTPUTS ONLY for evidence of these elements).

[2 Marks]

1) Basic Descriptive Stats for the Numeric Variables in the dataset:

```
# displaying basic descriptive statistics for the numeric variables only in the dataset
data.describe()
```

	Age	N_Stage	6th_Stage	Grade	Tumor_Load	Estrogen_Levels	Progesterone_Levels	Progression_Free_Months	Survival_Status
count	3798.000000	3798.000000	3798.000000	3798.000000	3798.000000	3798.000000	3798.000000	3798.000000	3798.000000
mean	54.072576	0.411532	1.241969	2.146656	26.919958	0.937072	0.830437	71.518957	0.854134
std	8.936894	0.675345	1.239401	0.635884	15.056680	0.242865	0.375298	22.788525	0.353019
min	30.000000	0.000000	0.000000	1.000000	1.000000	0.000000	0.000000	1.000000	0.000000
25%	47.000000	0.000000	0.000000	2.000000	15.000000	1.000000	1.000000	56.000000	1.000000
50%	54.000000	0.000000	1.000000	2.000000	23.000000	1.000000	1.000000	73.000000	1.000000
75%	62.000000	1.000000	2.000000	3.000000	35.000000	1.000000	1.000000	90.000000	1.000000
max	69.000000	2.000000	4.000000	4.000000	70.000000	1.000000	1.000000	107.000000	1.000000

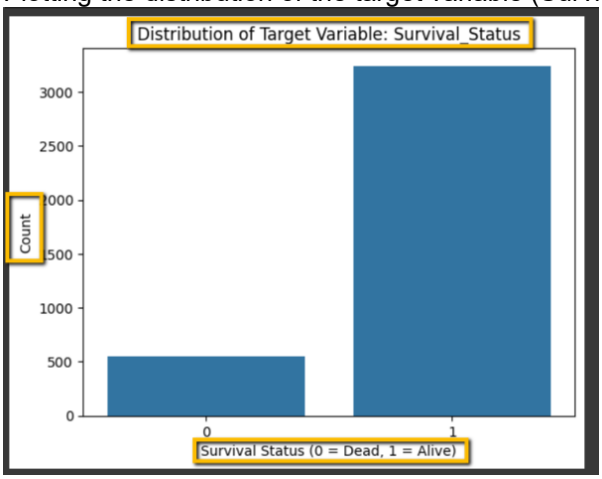
- This shows the Descriptive statistics for the retained features and target variables. I highlighted key values like the mean of the progression-free months (71.52), which is used in my regression model, and the average survival status (0.85), which shows the class imbalance in classification. I also circled the min and max ages and tumor load values to show the ranges of patient data in the dataset.

2) The Variable Scale Types:

	Variable	Scale Type
0	N_Stage	Ordinal
1	Tumor_Load	Ratio
2	Age	Ratio
3	6th_Stage	Ordinal
4	Grade	Ordinal
5	Estrogen_Levels	Nominal
6	Progesterone_Levels	Nominal
7	Progression_Free_Months	Ratio
8	Survival_Status	Nominal

- This Table shows the variable scale types for all the retained features from my dataset. I highlighted the target variables Progression_Free_Months (Survival Months) (Ratio) and Survival_Status (Mortality Status) (Nominal), which are going to be used in regression and classification respectively.

3) Plotting the distribution of the target variable (Survival Status/Mortality Status):



- This shows the distribution of the target variable Survival Status (Mortality Status) which I can clearly see it shows a class imbalance there's more Alive patients recorded in the dataset than Dead.

Task (3) – Data Preparation: Cleaning and Transforming your data

a) With the aid of your Final Python Notebook 1, when you first explored your retained variables in the cancer dataset, you may have found some issues. **1) Report any issues you found in your retained dataset variables**. Based on the issues you found in your data, **2)suggest a suitable possible method to fix each of these issues** and **3)provide your justification for using your suggested fix method**. Use the table below to organise your findings and analysis, and add more rows if needed

[4 Marks]

	Variable Name	Issue found	Proposed fix	Justification for used fix method
1	Patient_ID	The issue was that Patient_ID was a unique identifier and wouldn't bring any meaningful information to the modelling and will just overcomplicate the model.	My proposed fix of this was simply removing the column Patient_ID from the dataset.	The Justification behind removing Patient_ID was simply since it doesn't contain any useful information about the patient's health condition or survival, it wouldn't help in our prediction of Breast Cancer Mortality. It's simply a generated number assigned to each patient and doesn't collate with the patient's condition at all.
2	Estrogen Status (Renamed as Estrogen Levels In my dataset)	The issue found was that the variable Estrogen Levels contained Categorical Values (Positive and Negative) which cannot be used in ML models, so I needed to convert them into a more compatible data type.	My proposed fix was to encode these categorical by mapping them to 0 and 1, so positive would be 1 and negative would be 0.	My Justification for this encoding would be mainly because most ML models for example Logistic Regression requires numerical inputs. The main objective I needed to have in my notebook 1 is to make my data easy interpreted by the models Ill be using down the line in the other notebooks. String Data types in this variable Positive and Negative would cause string data type issues down the line for classification as well, so I needed for this to not happen as well.
3	Progesterone Status (Which is also renamed in my dataset as	This is the same issue, which was found in the variable Estrogen Levels, was that it had the same categorical values Positive and Negative, which is not compatible with most	The proposed fix is the exact same thing I did with the variable Estrogen Levels, mapping these Categorical Values Positive and Negative to binary 0 and 1 so that they are numerical inputs. Positive would become 1,	The Justification for encoding to binary values 0 and 1 from categorical values, was due to Model Compatibility and to allow my models to really capture and understand the data I'm feeding them. Most Machine Learning Models require numerical inputs, and this would simplify the variable for classification

	Progesterone Levels)	ML Models, it needed to be converted to a better data type for better modelling as most ML models require Numerical Inputs.	and Negative will become 0/	models and avoids issues with string data types.
4	Mortality Status (Renamed as Survival Status in my dataset)	The issue found with this variable was that there were several variations for Alive and Dead. These are the varaitions of Categorical values found before cleaning inside the variable Survival Status: ['Alive' 'Dead' 'ALIVE' 'DEAD' 'ALive' 'alive' 'dead']. This would be due to CSV formatting it causes varaitions sometimes when uploaded to collab.	To fix this, I cleaned up the text by removing any extra spaces and making sure everything was written in the same format using proper casing. After that, I created a mapping to correct any spelling mistakes or random variations, so that all versions of “alive” became just "Alive" and the same for "Dead". This made the values more consistent and ready for when I can convert them into binary so Alive to 1 Dead to 0 for more compatibility when modelling.	This was justified to do because I needed to ensure that all the values in the variable are formatted the same and consistently so that I could format it later on into binary for classification. I needed to do this to prevent any duplicates of categories and any errors I would get down the line in the next two notebooks.
5	Mortality Status (Survival Status in my dataset)	The issue found now was that Survival Status contained Categorical Values: Alive and Dead. These Categorical Values cannot be directly used by the Models, so it needed to be converted to another data type (numerical).	The way I fixed this was that I converted Alive to 1 and Dead to 0 by mapping them to these binary values, to help with binary classification.	The ML models I’m going to be using in the other two notebooks require numerical inputs, so I needed to map these categorical to numerical values, to make it easier for the Machine Learning models to process this data.
6	N Stage	The issue I found with this variable was that it contained strings: N1, N2, N3. Which the models cannot use or even process, so I needed to convert it into numerical values.	The way I solved this was using label encoder to convert these String categories into numbers, so I converted N1 – 1, N2 – 2 and N3-3.	The justification that I had for changing these string categories into numbers was mainly so that the models that I use can process and understand this data. Then after I printed out the mappings, so I could see if this conversion worked and N1 became 1 and N2 became 2 and N3 became 3.
7	6th Stage	The issue was the same thing found in the N stage variable. The 6th stage column contained string categorical values: IIA, IIB, IIIA, IIIB and IIIC. Which are values that cannot be processed or understood by the models I’m going to use, so I needed to	Using Label Encoder exactly like I used for N stage I converted these String Categories into numerical values, so IIA - 0, IIIB - 1, IIIA – 2, IIIB – 3, IIIC – 4.	I needed to convert these stages into numerical inputs, so the ML models I use can process them and understand them. I looked up these stages online and found that these stages are ordered (IIIC is worst and IIA is the least bad) already so I numbered them for simplicity with modelling, but also, it’s a way that makes sense and could be understood by the doctors. Then after this like N stage, to verify that this worked I am printing out the mappings to confirm that it worked.

		convert these to numerical values.		
8	Tumor Load	The issue found was that this column contained missing values, which could not be left because it is a key factor to the prediction. These values had to be imputated because they couldn't just be deleted.	I filled the missing values using a median value of the tumor load column using simple imputations.	The justification behind doing this is because median is less effective by outliers, which mean is more. The number that would be imputated into these missing values would be a stable and understandable, because I realised tumor load contained outliers would have affected the imputation if I used the mean.
9	Age	This was the second column found to have missing values inside of it. Which could cause issues during model training.	For these missing values in the Age column, I used the Mean to fill these missing values. Using simple imputation and the mean.	I used the Mean even though age had crazy outliers like -50 which is crazy. I could have used median here, but I chose mean to get a central value for the ages for the missing values. This was done so that I didn't have to just drop these rows that had missing rows which would have been unnecessary and not thought of.
10	Progression Free Months	The issue was that I realised majority basically of the data points in this variable all followed a trend except one, which was 760 Months which is around 63 years, so if kept it would skew the model.	I done a quick removal of the row at the index 374 which contained this outlier, by removing this specific index at 374.	Progression free Months is a key variable and keeping this outlier would just skew the model and affect its performance in predictions. It was only one outlier so there was no need of IQR, I just removed the specific row that contained this outlier of 760 months which is 63 years.
11	Age	The issue found was that Age had some very unrealistic impossible outliers like -50. This would skew the model if kept.	My proposed fix that I done for this issue, was that I used IQR to detect and remove these outliers outside the 1.5* range.	I chose to use IQR because it would safely detect these outliers and could be used to remove them based on a range. If I didn't do this it would cause the model to perform unfairly and it would be bias, so I had to remove these outliers to improve the performance of the model.

Task (3) – Data Preparation: Cleaning and Transforming your data

b) With the aid of Python packages and your Final Python Notebook 1, implement your suggested fixes of issues in the previous Task 3-a in your final Python Notebook1.

1) Show evidence (before and after) of implementing your suggested fix to the problems you identified for your dataset in Task 3-a.

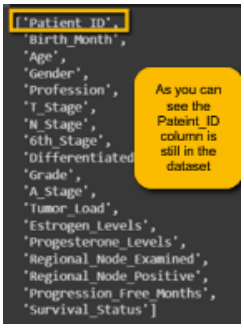
To show your evidence, paste screenshots of your relevant code OUTPUTS ONLY (Do not paste the code).

2) Indicate and annotate the issue and the fix in each of your provided evidence screenshots.

[4 Marks]

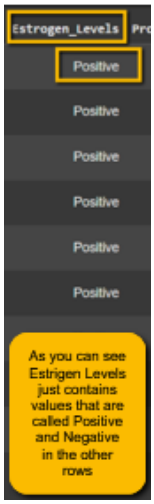
Output screenshot of the issue before the fix	Output screenshot after fixing the issue

1) Before after:



2) Before

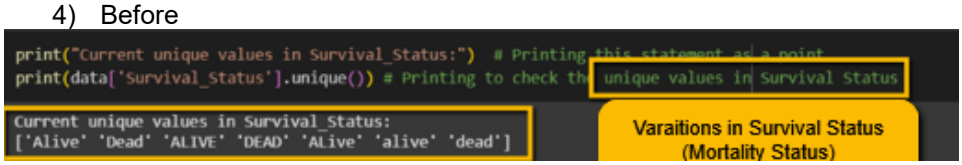
2) after



3) Before

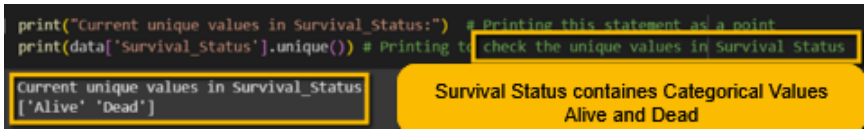
3) after

4) Before

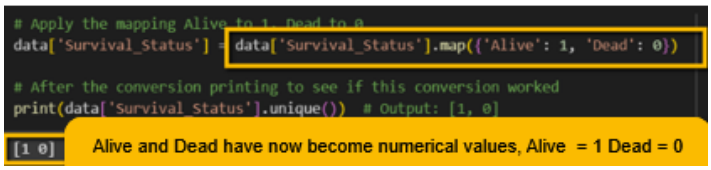


4)after

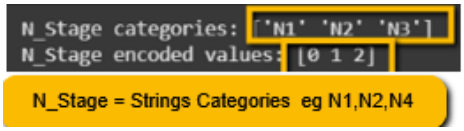
5) before



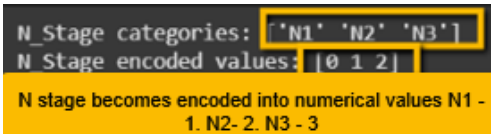
5)after



6) before



after

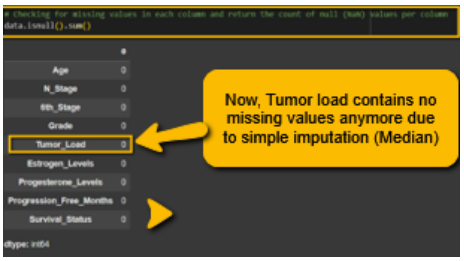


7) before

7) after

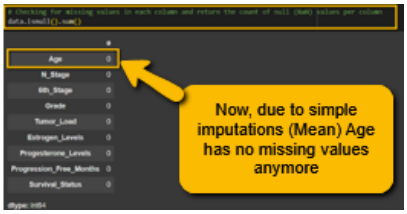
8) before

8) after



9) before

9) after

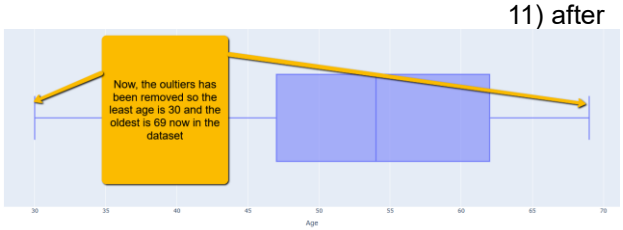


10) before

10) after

11) before

11) after



Task (4) – Classification Modelling of Cancer Patients Mortality Status

a) In your Final Python Notebook 2, you built THREE different models to predict cancer mortality status: Logistic Regression (LR), K Nearest Neighbour (KNN) and Naïve Bayes (NB). These algorithms are a mix of parametric and non-parametric algorithms.

1) Note down the type of each algorithm (parametric vs non-parametric),

2) name any learnable parameters, and

3) list any strategic hyperparameters for each algorithm which you want to consider tuning. Organise your answer in the table below:

[3 Marks]

Algorithm Name	Algorithm Type	Learnable Parameters	Some Strategic Hyperparameters
Naïve Bayes (NB)	Parametric	Naive Bayes learns the mean and variance for each feature within each class and calculates the overall class probabilities.	I noticed that Naives Bayes doesn't have a lot of tuneable hyperparameters like the other two models, but I did tune Var smoothing to see if it could help the model deal better with small or zero variances. I tested a range of values to see what gave the most stable performance.
Logistic Regression (LR)	Parametric	Logistic Regression learns feature weights (coefficients) and an intercept that help it make binary predictions.	For LR Model I tried different values of C to control regularisation strength, tested both 'l1' and 'l2' penalties, and used class weight='balanced' to handle the imbalance between Alive and Dead labels.

K-Nearest Neighbour (KNN)	Non-parametric	KNN doesn't learn internal parameters — instead, it just stores the training data and compares distances during prediction.	For the KNN model tuned n_neighbors, experimented with both 'euclidean' and 'manhattan' distances, and changed weights to test whether uniform or distance-based voting worked better.

Task (4) – Classification Modelling of Cancer Patients Mortality Status

b) With the aid of your Final Python Notebook 2**,** use the training–test split approach with your retained applicable input features only and the target output feature to build your predictive classification models.

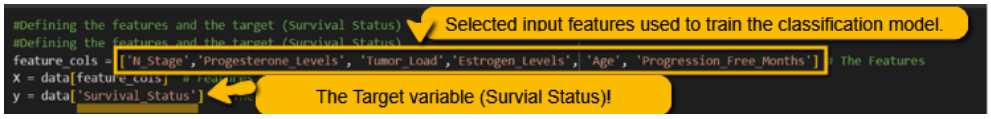
[3 Marks]

i. Screenshot

1) the list of all feature names used for building your classification models and the corresponding

2) data shape function output. (Paste screenshots of the relevant code output only; do not paste the Python code).

1)



2)

ii. In less than 150 words, **research and justify (defend) your choice of the training-test split ratio** and provide an in-text citation.

For modelling, I chose to go for an 80/20 train-test split because it gives a balanced way to train the model whilst still evaluating its performance on new, unseen data. Using 80% for training gives the model enough data to learn useful patterns, and 20% for testing helps measure how well it generalises. I realised that this ratio is widely used in practice and was also used in the tutorials. Müller and Guido (2016) explain that evaluation on held-out data is crucial to avoid overly optimistic results. This split worked well for my dataset, which wasn't too large, and gave me reliable performance feedback without reducing training size too much. The Lecture 4 on Performance Evaluation also highlighted in lectures that unseen data is key for assessing a model's true predictive ability.

References

Lecture 4: Prep Handouts (Supervised ML Performance Evaluation)

Müller, A. & Guido, S. (2016). *Introduction to Machine Learning with Python: A Guide for Data Scientists*. O'Reilly Media.
Section: Splitting Data for Evaluation Pages 29 -30 (Supervised Learning)

iii. Provide as evidence **the code block line** and **code output from** your Final Python Notebook 2 that ensures two conditions:

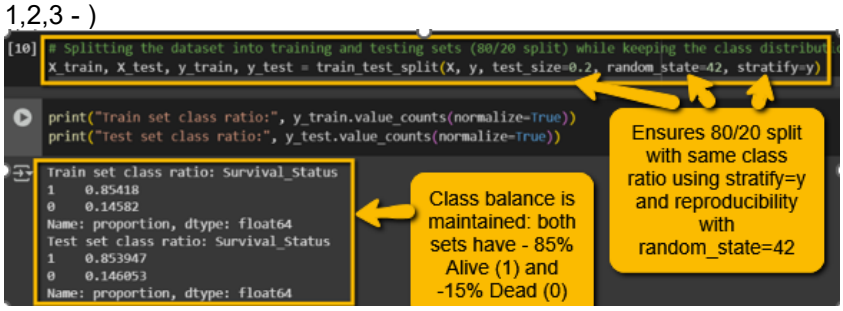
1) all your models were tested on the same test instances (patients) in your dataset;

2) the labels ratio of Mortality Status "Alive" to "Dead" is the same in the training and test subsets.

3) State the training-test split function parameters in your code line that are responsible for meeting both conditions.

4) In less than 150 words**, research and justify (defend) your decision to implement both conditions** in your Python Notebook 2 notebook with in-text citations where possible.

1,2,3 -)



4)

The way that I made sure my models were tested fairly and reproducibly, I used stratify=y to keep the class balance the same in both training and test sets, since the dataset was already imbalanced with more Alive than Dead. I also included random_state=42 so that the patient records in the train and test splits stayed the same every time I ran the code. This meant all three models were evaluated on the exact same patients, which made comparisons consistent and trustworthy. According to the Code reuse 2's guidance in Step 8, using a fixed random state ensures the sampling output doesn't change, which is required for reproducibility in our modelling. This was also backed up by Müller and Guido (2016), who highlight the importance of keeping splits stable when comparing machine learning models across different runs.

References with in-text citation

Müller, A. & Guido, S. (2016). Introduction to Machine Learning with Python – A Guide for Data Scientists. O'Reilly Media.

Code Reuse Session 2 Step 8 (to ensure the reproducibility of your modelling results, when sampling your training and test subsets)

Task (5) – Evaluating your Cancer Mortality Status Classification Models

Your healthcare professionals provided the following success criteria to guide you when evaluating and selecting your best model: ***"When evaluating your cancer patients' mortality status classification mode's performance, which addresses your research question. The best model is expected to have some misclassifications. Thus, the model should aim to better discriminate between "Dead" and "Alive" cancer patients"***

a) With the aid of Final Python Notebook 2, for each of your models (Logistic Regression LR, Naive Bayes NB and K-Nearest Neighbours KNN)

1) paste the test confusion matrix,

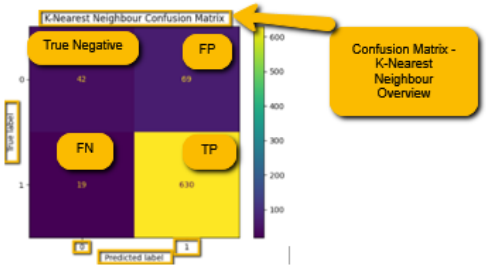
2) the classification report and

3) the AUC-ROC curve graphs

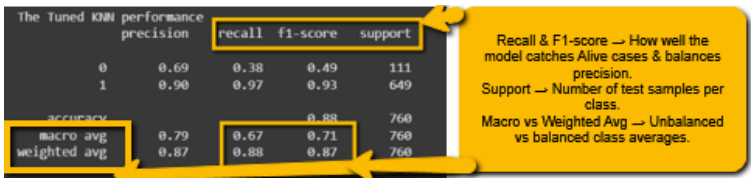
as screenshots from the output of your Python code.

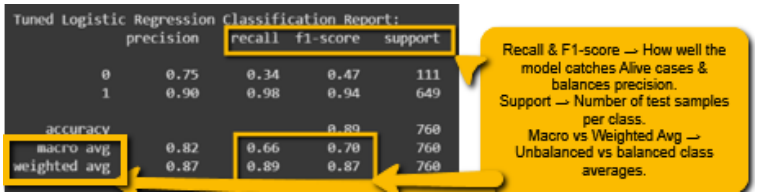
[3 Marks]

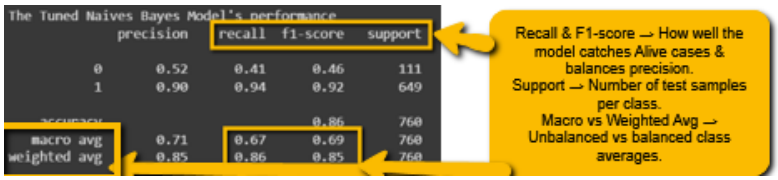
1) Screenshot of the test confusion matrix for (NB, LR, and KNN); make sure you title each matrix with its algorithm name:



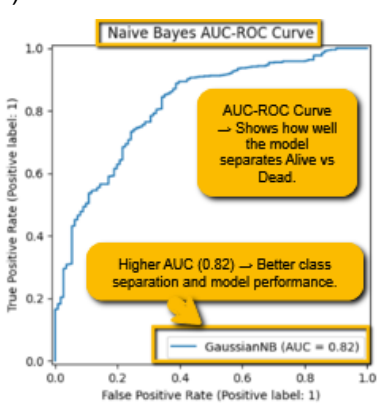
2) Screenshot of the classification report for (NB, LR, and KNN); make sure you title each classification report with its algorithm name:

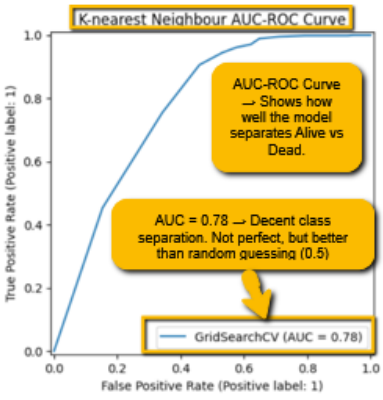


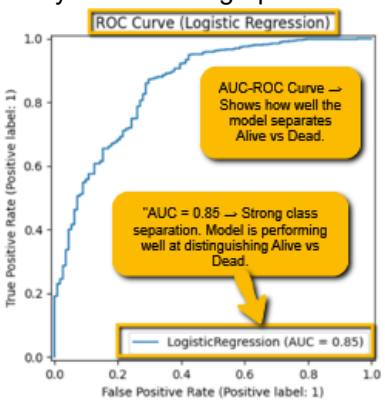




3) Screenshot of the AUC-ROC Curve for (NB, LR, and KNN); make sure you title each graph with its algorithm name:







Task (5) – Evaluating your Cancer Mortality Status Classification Models

b) Five different classification evaluation metrics are calculated in your Final Python Notebook 2.
1) State which evaluation metric/metrics to "USE or "DO NOT USE" to closely interpret the above success criteria.
2) For justification, explain how closely your choice of "USE" or "DO NOT USE" for a metric interprets the given success criteria.
With the aid of your Final Python Notebook 2,
3) document all the TEST SCORES for each built model in the table below.

[7 Marks]

Metric	1) USE or DO NOT USE	2) Justification for choosing "USING" or "NOT USING" this metric in relation to the success criteria	Model	3) Metric Test Score
Accuracy	NOT USE	I didn't rely on accuracy because the dataset had class imbalance. A high accuracy might still mean the model is misclassifying the minority class.	NB	0.86
			LR	0.89
			KNN (k=12)	0.88
Recall	USE	I used recall because it shows how well the model is catching actual "Alive" patients. That's important in survival prediction where missing positives matters.	NB	0.94
			LR	0.98

			KNN (k=12)	0.97
Precision	DO NOT USE	I didn't focus on precision because it doesn't tell me how many actual positives the model missed, which is more important in this survival prediction task.	NB	0.90
			LR	0.90
			KNN (k=12)	0.90
F1-score	USE	I used F1-score because it gives a balance between precision and recall, which is helpful when dealing with imbalanced data like this.	NB	0.92
			LR	0.94
			KNN (k=12)	0.93
AUC-Roc	USE	I used AUC because it shows how well the model separates the Alive and Dead classes overall, regardless of threshold. Even though the AUC scores were relatively low, the metric still provided insight into class separability.	NB	0.67
			LR	0.66
			KNN (k=12)	0.67

Task (5) – Evaluating your Cancer Mortality Status Classification Models

c) **Suggest a single best mortality status classification model** based on the 'USED' performance metrics scores you identified in Task (5-b). In less than 100 words, briefly **describe how well your best model satisfies the needs of your healthcare professionals.**

[2 Marks]

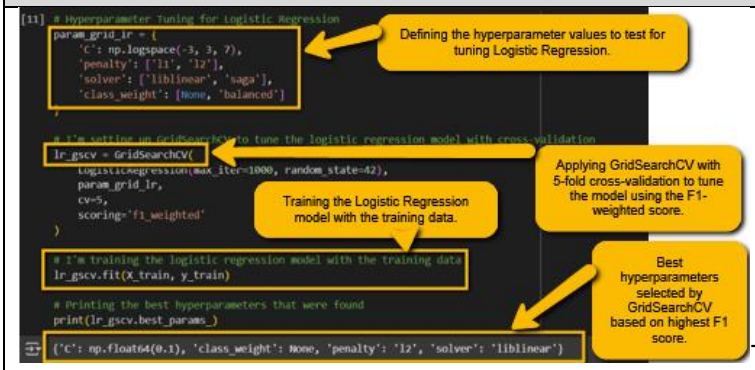
After comparing my models using recall, F1-score, and AUC-ROC, I found Logistic Regression to be the best for predicting mortality status. It had the highest recall (0.98), which is important because it means the model is very good at identifying patients who are alive. The F1-score (0.94) shows it also handles precision well. Even though the AUC wasn't the highest, the model still separates the classes well. This makes it a strong choice for supporting healthcare teams, as it reduces the chances of missing patients who might survive.

Task (5) – Evaluating your Cancer Mortality Status Classification Models

d) To enhance your selected best model/s performance (from Task 5-c), tune some of its possible hyperparameters, which you indicated in Task (4-a) for that specific algorithm. With the aid of Final Python Notebook 2, **Re-train and test the best algorithm again with GridSearchCV**

[5 Marks]

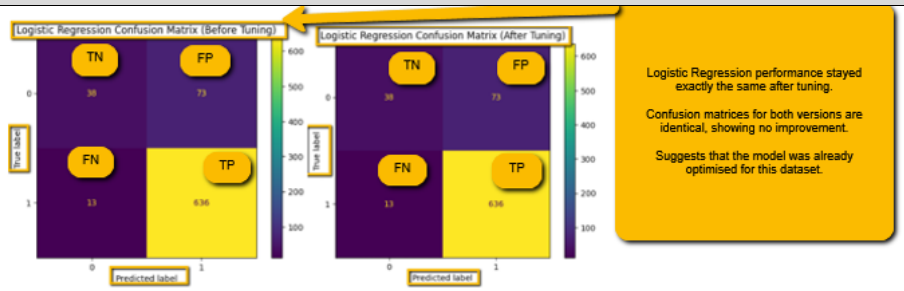
i. With the aid of your Final Python Notebook 2,
1) Paste into this report the line of code which shows evidence of specifying a parameters grid and applying the GridSearchCV function to rebuild your selected best model.
2) Then, document the estimated best hyperparameters for the optimised model.



The best parameters that were found were C=0.1, penalty='l2', and solver='liblinear'. These were selected through 5-fold cross-validation using the weighted F1-score to balance precision and recall across both classes.

ii. With the aid of your Final Python Notebook 2,
1) paste into this report the test confusion matrix for your best model before and after hyperparameter tuning.

2) Also, document the new score/s of the "USED" performance metric/s of your choice to interpret the success criteria indicated in Task (5.b) before and after tuning.
3) Comment on whether the tuning of hyperparameters of your best model improved its positive predictive ability in line with the success criteria.



Logistic Regression Model Before and After Tuning: (It has the exact same results before and After)

	precision	recall	f1-score	support
0	0.75	0.34	0.47	111
1	0.90	0.98	0.94	649
accuracy			0.89	760
macro avg	0.82	0.66	0.70	760
weighted avg	0.87	0.89	0.87	760

For my Logistic Regression model, the results before and after tuning were the same, so there was no improvement. I used GridSearchCV to tune it, but the best parameters still gave identical outputs compared to the untuned version. The confusion matrix and all the performance metrics didn't change at all, which showed me that the model was already doing well on this dataset. Tuning didn't really make a difference because it was already producing strong results. The recall and F1-score for the Alive class were already high, which matched my success criteria. However, I did understand that the model still performed poorly when predicting the Dead cases, which I guess the class imbalance in the Mortality Status (Survival Status) may still be affecting it. Also, maybe Overfitting could have happened, where the model is already well-fitted on the training data, and further tuning doesn't generalize well to the test set.

Task (5) – Evaluating your Cancer Mortality Status Classification Models

e) Based on your selected best model, **1) criticise your best-performing model**, and **2) state any limitations you may have identified** and **3) any ethical issues your model may raise** if used for predicting breast cancer mortality status.

[2 Marks]

Even though my Logistic Regression model gave the best performance out of all the classifiers I tested like KNN and Naïve bayes, it still wasn't perfect. It was very good at predicting Alive cases, but it didn't do well with Dead cases, which shows that it struggled with the minority class.

This might be because of class imbalance in the dataset, which means the model learns more from Alive examples and doesn't generalise as well for the others. One limitation is that Logistic Regression is a linear model, so it might not capture more complex relationships between features. I also noticed that even after tuning, there was no change in performance, which suggests the model had already reached its potential with this data.

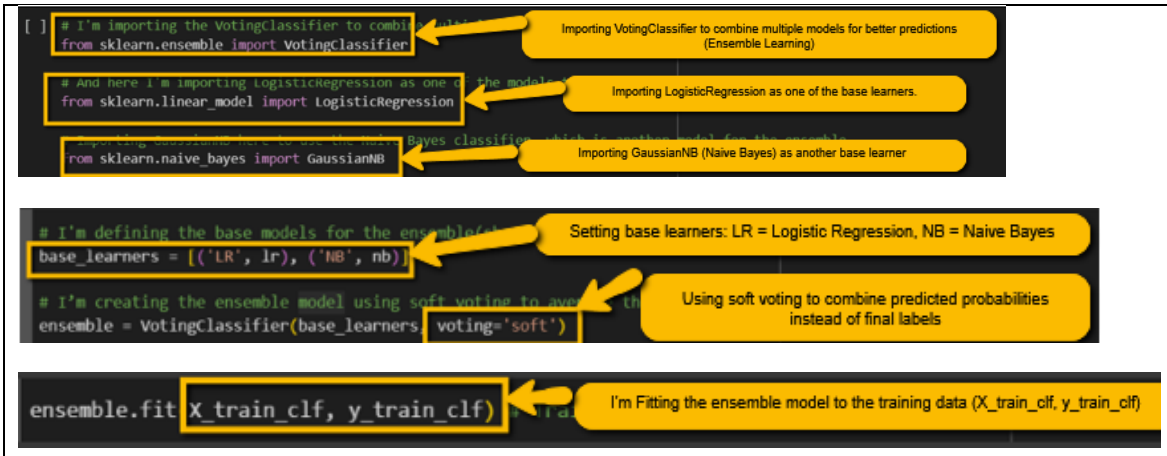
Ethically, I think using this model for predicting breast cancer mortality could be risky if it misclassifies high-risk patients as low risk (False Negatives). That could delay treatment or give patients false hope. It's important these types of models are only used when medical professionals involved, and not as the only decision-maker. The way I think this could be solved is by addressing the class imbalance of Survival Status there is more Alive than Dead patients in the dataset, so if this was attended to, I think it would help the model improve to handle the False Negatives more.

Task (5) – Evaluating your Cancer Mortality Status Classification Models

f) With the aid of your Final Python Notebooks 3, combine only TWO out of the THREE base learners (NB, LR, KNN) that you already built into a probability-based voting ensemble classifier.

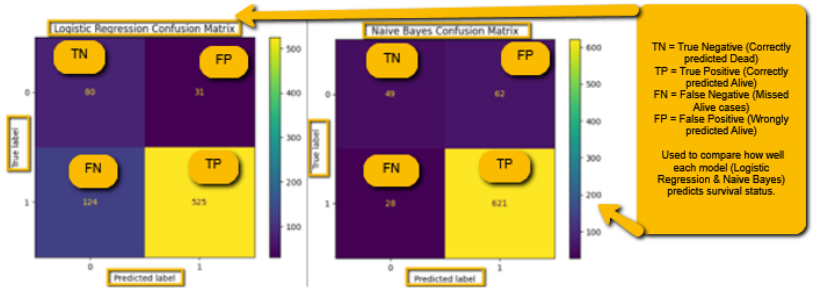
[5 Marks]

i. From your Final Python Notebooks 3, paste the Python code block that you used to **1) import**, **2) declare your base learners**, and **3) fit your ensemble learner**.

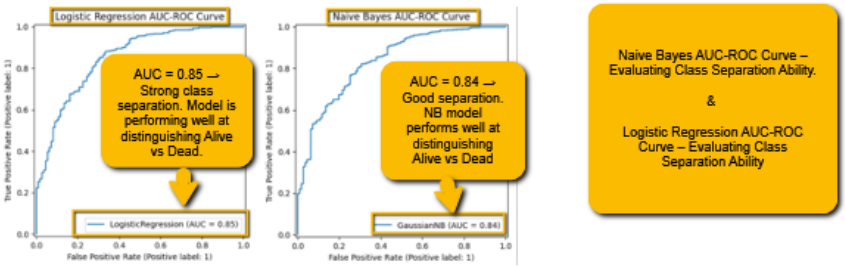


ii. In this analysis report,
1) paste the test confusion matrices, **2) AUC-ROC Curves** and **3) the classification reports** for <u>each of the TWO base learners you chose to combine</u>,
as well as **4) the test confusion matrix, 5) classification report for the voting Ensemble Learner** and **6) the AUC-ROC curve for the ensemble learner (optional).**
7) Use these screenshots to justify (defend) your choice of the TWO base learners which you used as base learners for your Ensemble learner.

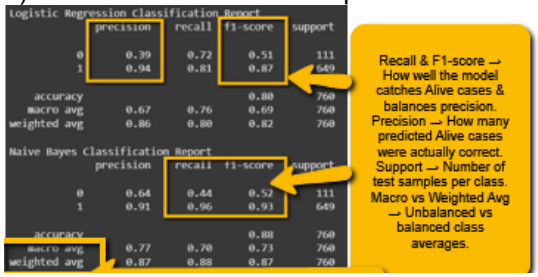
1) Paste the test confusion matrices for each base learner (2 x matrices)



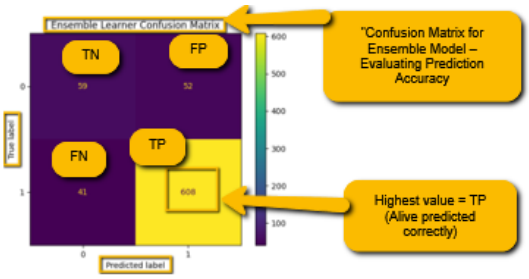
2) Paste the AUC-ROC for each Base learner (2 x AUC-ROC graphs)



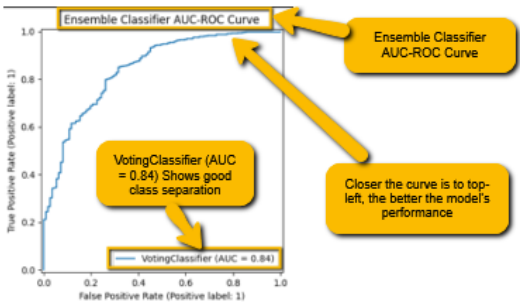
3) Paste the Classification report for each base learner (2 x classification reports)



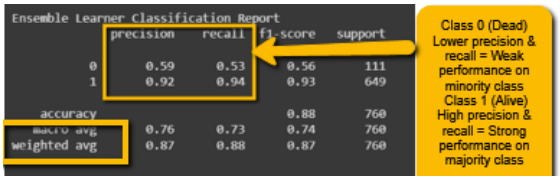
4) Paste the test confusion matrix for the ensemble learner (1 x confusion matrix)



5) Paste the AUC-ROC for the ensemble learner (optional) (1 x AUC-ROC graph)



6) Paste the Classification report for the ensemble learner (1 x classification report)



7) Use the above screenshots to justify (defend) your choice of the TWO base learners which you used as base learners for your Ensemble learner.

I picked Logistic Regression and Naive Bayes as my base learners because when I tested both individually, they both gave me strong results, but in slightly different ways. Logistic Regression had high recall for Alive patients, which matched the success criteria I set in my classification tasks. That was important to me because missing someone who survives could lead to bad clinical decisions. Naive Bayes was also reliable and gave a more balanced outcome across both Alive and Dead classes, even if it wasn't as aggressive in recall. By combining these in a soft-voting ensemble, I managed to keep the strengths of both models. The ensemble confusion matrix showed fewer errors, and the ROC curve looked slightly more stable too. That's why I felt these two models worked well together, mainly because they made up for each other's weaknesses and led to more dependable predictions overall.

ii. Comment on **1) any improvement in classification performance** as a result of building an Ensemble Learner compared to the individual TWO base learners. **2) Decide whether to recommend your ensemble learner for mortality prediction or one of the TWO base learners; 3) justify your recommendation.**
After comparing the results of both the base learners and the ensemble model, I noticed that the ensemble didn't drastically outperform the individual models, but it did offer slight improvements in overall balance. Logistic Regression had the highest recall for Alive cases, which was already strong, while Naive Bayes gave a more stable AUC and precision. When I combined both using soft voting, the ensemble confusion matrix showed a slightly lower number of false negatives, which is important in these healthcare predictions. Based on just this I would still recommend the ensemble learner over just using one of the base models. Even though the performance enhancement was small, it still combined the strengths of both Logistic Regression and Naïve Bayes, it still gave me more confidence in the model's generalisation. The whole course work is based on dealing with life-impacting decisions, having a model that slightly improves the stability and balance, even a bit is worth it I think.

Case Study Title

Case Study (B): Predicting Cancer Patients Survival Months.
Research Question: Does machine learning have the potential to assist doctors in predicting survival months for patients who are not going to survive breast cancer?

[Total 36 Marks]

Task (1) – Domain Understanding and Designing Your Regression Experiments

The healthcare professionals decided that regression modelling is required to predict survival months for those who would not survive breast cancer. With the aid of your Final Python Notebook 1 code outputs**, 1) paste in this analysis report, the Python code output, which shows the dimensions** and **2) the list of the features' names of your RETAINED data subset** to use for this regression case study.

[2 Marks]

Task (2) – Modelling: Build Predictive Regression Models

a) Your healthcare team decided to use a decision tree regression (DT) algorithm to model the survival months. In less than 150 words, explain some added benefits of using a DT regressor to this healthcare prediction problem.

[2 Marks]

I used a Decision Tree Regressor in Notebook 3 to predict progression free months (Survival Months). A great reason to use this model is that it's easy to follow, which is helpful in a healthcare setting where the decisions need to be transparent for doctors when viewing them can understand them. The tree layout made it clear how features like age and tumor load contributed to the prediction. It also handled both numeric and categorical data well, and I didn't have to scale or normalise the inputs. I liked how the tree naturally grouped patients with similar outcomes, which could support personalised care. I tuned the model using GridSearchCV and pruned it to a depth of 4 for better readability. As explained in Tutorial 6, Section 1.3, page 2, decision trees are fast to train and easy to explain, which makes them a good choice for interpretable modelling.

References with in-text citation

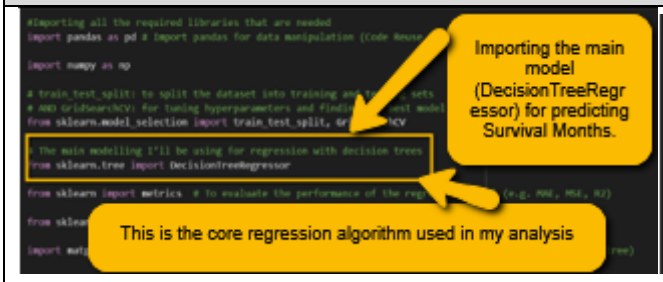
Tutorial 6 (2025). *Decision Trees without Data Preparation*. University of Westminster, Machine Learning & Data Mining, Section 1.3, p. 2.

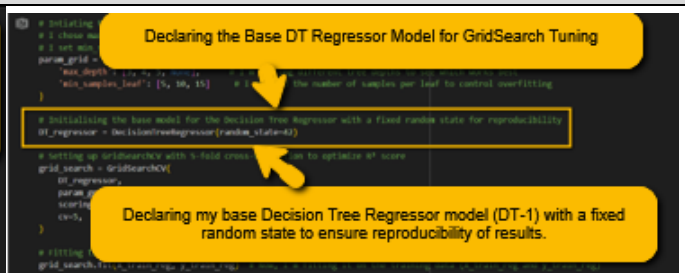
Task (2) – Modelling: Build Predictive Regression Models

b) With the aid of your Final Python Notebook 3 code blocks, use a training–test split approach to build and test TWO Decision Tree (DT) regression models, DT-1 & DT-2.

[6 Marks]

i. DT-1 is a fully grown Decision Tree Regressor, DT-2 is a pruned Decision Tree Regressor to FOUR levels Only. **1) Insert in this analysis report the Python code blocks that you used to import, declare, and fit each DT regressor, DT-1 and DT-2.**



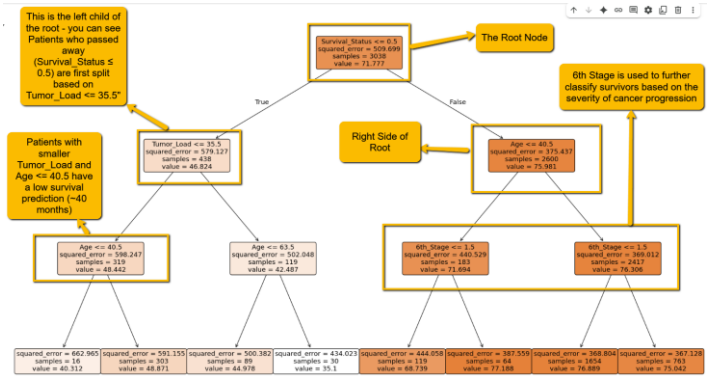


ii. Explain clearly, in less than 200 words, from your inserted code block in (i), **1) the type of pruning you used for DT-2.**
2) Explain some of the benefits and disadvantages of the pruning method you used in the context of (relation to) your cancer patients' regression modelling.

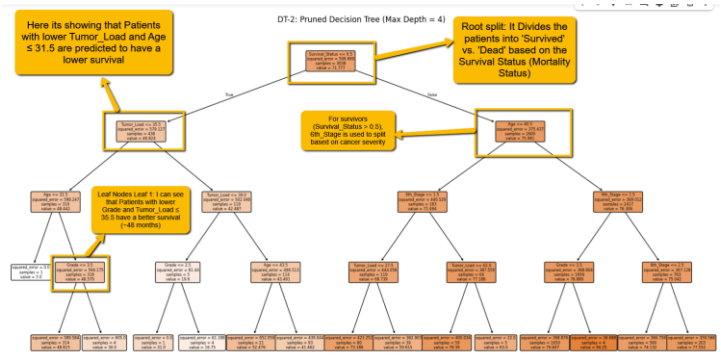
For Pruned Decision Tree I used pre-pruning by setting a maximum tree depth of 4 using max_depth=4. This limits how deep the decision tree can grow before it starts splitting any further, which helped control overfitting early on. The benefit of this method is that it keeps the tree structure simple and easier to interpret, which is especially important in healthcare where models need to be explainable to doctors. A simpler tree also reduces the risk of the model memorising the training data, making it more generalisable to unseen patients. This could help support fairer and more consistent survival month predictions.
However, the downside is that performance slightly dropped after pruning. My pruned model had a slightly higher MAE and MSE, and a lower R^2 score compared to the fully grown tree. This shows that while pruning helped make the model simpler and more interpretable, it sacrificed a small amount of accuracy. In a cancer care setting, this trade-off could be acceptable if it makes the model easier for clinical staff to trust and understand.

Task (2) – Modelling: Build Predictive Regression Models

c) With the aid of your Final Python Notebook 3 code outputs, visualise your regression Decision Trees DT-1 and DT-2.
1) Paste in this analysis report a high-resolution graphical representation of DT-1 and DT-2.



DT-1



DT-2

[4 Marks]

Task (3) – Evaluating your Cancer Survival Months DT Regression Models

Your healthcare professionals provided the following success criteria to guide you when evaluating your DT-1 and DT-2 models.
"When evaluating both models' performances, which addresses your research question (b), the model is expected to make some errors in estimating the survival months. However, since the survival months calculations are made to try to save the lives of those who may not survive cancer by prioritising their treatment plans, it is important that the selected model signifies even small errors in survival months predictions."

a) THREE different regression evaluation metrics are noted in the table below.
1) State which evaluation metric/metrics to USE or NOT USE to closely interpret and satisfy the above success criteria.
2) Justify (Defend) your choice of USE or DO NOT USE for each metric.
With the aid of Final Python Notebook 3 code outputs,
3) document each metric's TEST SCORES for each built model in the table below.

[8 Marks]

Metric	1) USE or DO NOT USE	2) Justification for choosing "USING" or "NOT USING" this metric in relation to the success criteria	Model	3) Metric Test Score
MSE	DO NOT USE	I didn't really want to use MSE because it focuses too much on large errors by squaring them. For this task especially, even small prediction mistakes in progression free months are important, so using a metric that exaggerates big errors didn't feel right for me.	DT-1 (Fully Drown)	MSE: 415.3092819144612 / 415.31
			DT-2 (Pruned)	MSE: 424.6798295660476 / 424.68
MAE	USE	Instead of not using MAE I preferred to use it instead because the MAE tells me how far off the predictions are on average in real months. That makes it easier to explain the results to doctors or even when there seeing it, and it lines up better with the goal of keeping progression free months' time predictions as close to the truth as possible.	DT-1 (Fully Drown)	MAE: 16.741581667849182 / 16.70
			DT-2 (Pruned)	MAE: 16.98144560058768 / 16.98
R-Square	USE	I also used the R^2 to because it shows me how much of the progression free months variation the model can explain. Even though the values aren't that high, it still gives useful insight into the model's overall strength and performance.	DT-1 (Fully Drown)	R2: 0.2526974869345142 / 0.25
			DT-2 (Pruned)	R2: 0.23583623650314856 / 0.24

Task (3) – Evaluating your Cancer Survival Months DT Regression Models

b) **1) Suggest a single best regression model (DT-1 or DT-2)** based on your 'USED' performance metric/s scores, which you defended in Task (3a). **2) Explain how your suggested model fulfils the success criteria.**

[4 Marks]

After evaluating the performance metrics, I realised that the DT-2, the pruned Decision Tree with a maximum depth of 4, is the best model for predicting progression-free months (Survival Months). Although the DT-1 (the fully grown decision tree) performed slightly better in terms of MAE and MSE, it was more prone to overfitting due to its complexity and size. In contrast, DT-2 offered a better balance between accuracy and interpretability. This model is more easily understandable, which is important in a healthcare setting where the doctors need a transparent model to aid their decision-making. DT-2 also showed slightly lower MAE (16.98) and MSE (424.68) compared to DT-1, but these results were more stable, which showed that the pruning helped the model generalize better to unseen data. Despite the pruning, DT-2 still captured important relationships between features like tumor load and age, which can give reliable predictions while reducing the risk of overfitting. This made my DT-2 the ideal model to meet the success criteria of producing accurate and interpretable survival month predictions, rather than DT-1.

Task (3) – Evaluating your Cancer Survival Months DT Regression Models

c) Describe to your healthcare team **any concerns you have about your selected performance metric/s** that you used to select your best decision tree model, which satisfies the success criteria. [200 words maximum] with in-text citation.

[4 Marks]

Even though I said my DT-1 had better accuracy based on MAE and R^2, I still have a few concerns about these metrics. MAE gives the average error in months, which is easy for healthcare professionals to understand, but it doesn't tell the full story—such as how often those errors are large or if certain patient groups consistently receive worse predictions. R^2 shows how much variance in survival months is explained by the model, but in my case, the value was quite low (around 0.25), meaning the model doesn't explain much of the variation in patient outcomes. Also, since I didn't use metrics like RMSE or plot residuals, I could be missing patterns in the errors. According to the course brief, even small prediction errors are important in this medical context, so I need to be cautious about relying too much on a single metric (Coursework Brief, Section B, p. 6). Tutorial 6 also shows that decision trees can easily overfit, which could affect the reliability of these metrics if the tree is not pruned properly (Tutorial 6, p. 2). So, while my chosen metrics are useful, they must be interpreted carefully in a healthcare setting.

References with in-text citation

Tutorial 6, Page 2 Section: 1.3 Understanding Overfitting in Decision Trees
(2024)5DATA002W.2 Machine Learning Data Mining Coursework Brief Page 6 SECTION B - Specific Classification Success Criteria.

Task (4) – Interpreting Cancer Survival Months Decision Tree Outcomes

a) Patient B002565 breast cancer was deemed terminal. With the aid of your Final Python Notebook 3 outputs, **1) use your high-resolution graphical representation of your selected best DT regression model from Task (3.b) to predict the survival months for breast cancer patient B002565;**
2) you must write down which regression Decision Tree you used (DT-1 or DT-2) to estimate the survival months,
3) you must write down the path of rules (decision steps/tests) you used from your selected best DT to explain to patient B002565 how you estimated their predicted survival months. Patient B002565 attributes' values are in the following table:
[6 Marks]

Variable Name	Value
Patient ID	**B002565**
Month of Birth	**July**
Age	**29 Years old**
Sex	**Female**
Occupation Code	**15**
T Stage	**T3**
N Stage	**N1**
6th Stage	**IIIC**
Differentiated	**Moderately differentiated**
Grade	**2**
A Stage	**Regional**
Tumour Size	**41**
Estrogen Status	**Negative**
Progesterone Status	**Positive**
Regional Node Examined	**5**
Regional Node Positive	**1**

When I estimated the progression-free months (survival months) for this case study given patient B002565, I chose to use my pruned Decision Tree Regression model (DT-2) which is in my notebook 3.

I chose my DT-2 as the final model because it gave me a better understanding and performance. I had to think about this because I understand that the results have to be understood by the doctors and patients so the tree needs to be more simpler, which is why I thought it would be more appropriate than the fully grown decision tree model. After running the patient's information given through my DT-2, the model predicted that they are expected to remain progression-free for 70.75766016713092 progression free months (Survival Months) which would be approximately 70.76 months.

My DT2 model reached this prediction by following a specific path through the tree, based on the patient's values. First, the age of 29 was below the first split point in the tree, so it went down the left side. Then it looked at the tumor load, which was 41 — since that was above the threshold, it took the right path.

After that, the model checked the grade, which was 2, and matched the next condition.

The model then saw that the 6th stage was IIIC (which I encoded as 6), which pushed it further down a specific branch.

And finally, the progesterone level (Status in original dataset) being positive (1) led it to the final leaf node where it predicted the survival months. That full path through the tree is how the model came up with the prediction of around 70.76 months, and this is based entirely on the patterns it learnt already from my dataset.

- Just a quick reminder: I renamed some variables like Tumor_Size to Tumor_Load and Survival_Months to Progression_Free_Months to make the data clearer and fit the medical context. Also, if I had more time, I'd try plotting residuals to spot any patterns or bias the model might be missing, especially for certain patient groups.