INFORMATICS INSTITUTE OF TECHNOLOGY

In Collaboration with

UNIVERSITY OF WESTMINSTER

Breast Cancer Mortality Prediction using Machine Learning

Module Name – Machine Learning and Data Mining

Module Code - 5DATA001C.2

Module Leader - Mr. Nipuna Senanayake

Student details:

Robosriyas Stravakske Fernando

W2082277 | 20232931

CS-20

May 2025

Table of Contents

List of Tables

List of Figures

Case Study (A) Analyses Report for Predicting Mortality Status Tasks

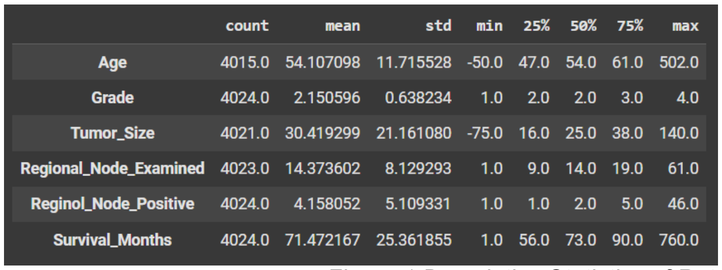
Task (1) – Domain Understanding: Classification

Variable Name	RETAIN or DROP	Brief justification for retention or dropping
Patient ID	DROP	This is just a unique value to identify the patients and has no predictive value for mortality status
Month of Birth	DROP	Month of birth likely has minimal relevance to cancer mortality and may introduce noise. Since age is already included, it adds no additional value to the model.
Age	RETAIN	Age is a proven independent predictor of breast cancer outcomes, as older patients typically face worse survival due to comorbidities and treatment differences.
Sex	RETAIN	Breast cancer behavior can differ between males and females, with males having around 85 deaths yearly compared to 11,400 female deaths (2017-2019).
Occupation	DROP	95% of this feature is missing on the dataset, and it is unlikely to have strong correlation with mortality status
T Stage	RETAIN	T stage (T1–T4) reflects tumor size and is a key part of cancer staging.
N Stage	RETAIN	N stage (N1–N3) shows lymph node involvement.
6th Stage	RETAIN	The BI-RADS system provides standardized assessment that supports staging and treatment planning.
Differentiated	RETAIN	Cell differentiation shows how abnormal and aggressive cancer cells are.
Grade	RETAIN	The Nottingham Grading System standardizes cancer aggressiveness assessment.
A Stage	RETAIN	indicates regional or distant cancer spread, distant metastasis (e.g., to lungs, liver, or bones) greatly raises mortality risk.
Tumour Size	RETAIN	Tumor size in millimeters gives a precise measure of cancer severity.
Estrogen Status	RETAIN	Estrogen receptor status guides treatment and affects prognosis, with hormone receptor-positive cancers generally showing better survival than hormone-negative ones.
Progesterone Status	RETAIN	Progesterone receptor status, like estrogen status, influences treatment and outcomes, offering valuable insights into tumor behavior.
Regional Node Examined	RETAIN	The number of lymph nodes examined affects staging accuracy and treatment planning.
Regional Node Positive	RETAIN	The number of cancer-positive lymph nodes directly indicates spread.
Survival Months	DROP	Survival months would create data leakage, as this information would not be available at the time of prediction in a real-world scenario.
Mortality Status	RETAIN	The target variable the model is about to predict.

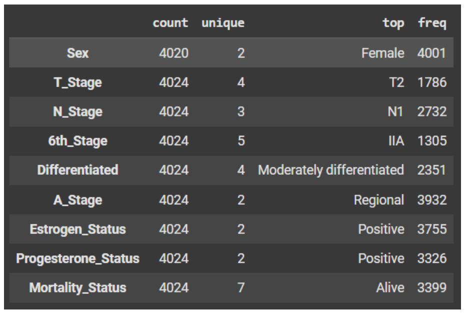
Table 1 Variable Analysis for Mortality Prediction

Task (2) – Exploring and Understanding Your Dataset

Basic descriptive statistics of the retained and target variables were obtained immediately after dropping the intended variables. Note that *survival months* appears as an object type because it's retained for the regression task and only removed for the classification task.



	count	mean	std	min	25%	50%	75%	max
Age	4015.0	54.107098	11.715528	-50.0	47.0	54.0	61.0	502.0
Grade	4024.0	2.150596	0.638234	1.0	2.0	2.0	3.0	4.0
Tumor_Size	4021.0	30.419299	21.161080	-75.0	16.0	25.0	38.0	140.0
Regional_Node_Examined	4023.0	14.373602	8.129293	1.0	9.0	14.0	19.0	61.0
Reginol_Node_Positive	4024.0	4.158052	5.109331	1.0	1.0	2.0	5.0	46.0
Survival_Months	4024.0	71.472167	25.361855	1.0	56.0	73.0	90.0	760.0



	count	unique	top	freq
Sex	4020	2	Female	4001
T_Stage	4024	4	T2	1786
N_Stage	4024	3	N1	2732
6th_Stage	4024	5	IIA	1305
Differentiated	4024	4	Moderately differentiated	2351
A_Stage	4024	2	Regional	3932
Estrogen_Status	4024	2	Positive	3755
Progesterone_Status	4024	2	Positive	3326
Mortality_Status	4024	7	Alive	3399

Figure 1 Descriptive Statistics of Retained Variables

Variable Name	Scale Type
Age	Ratio
Sex	Nominal (Binary)
T Stage	Ordinal
N_Stage	Ordinal
6th Stage	Ordinal
Differentiated	Ordinal
Grade	Ordinal
A Stage	Nominal (Binary)
Tumor Size	Ratio
Estrogen Status	Nominal (Binary)
Progesterone Status	Nominal (Binary)
Regional Node Examined	Nominal (Binary)
Regional Node Positive	Ratio
Mortality Status	Nominal (Binary)

Table 2 Types of Variable Scales

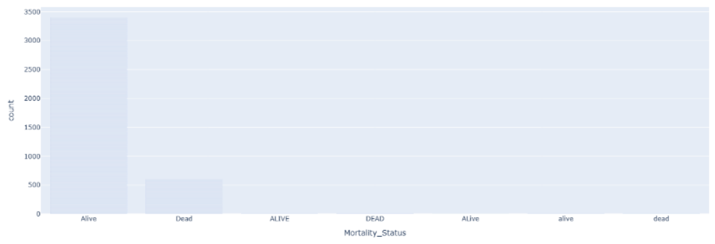


Figure 2 Distribution of target variable on the dataset.

Note that the target variable exhibits inconsistencies in its labels and should be appropriately encoded during data preprocessing.

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

a) Issues in the cancer dataset with suggested fixes and justifications in tabular form.

Variable Name	Issue found	Proposed fix	Justification for used fix method
Age	Contains extreme outliers (e.g., -50, 180, 500) that are practically not possible.	Dropping the records with extreme outliers.	These represent impossible biological values rather than natural variability and including them would compromise the clinical validity of the breast cancer mortality model.
Age	Certain values are missing.	Doing simple median imputation for the missing values.	Median imputation handles skewness, preserves central tendency, and resists outlier influence.
Sex	Certain values are missing.	Doing simple mode imputation for the missing values.	As a categorical variable, missing entries are filled with the mode (Female) since breast cancer mainly affects women.
Tumor Size	Contains a negative value, which is not practically possible for tumor size.	Dropping the record with outliers.	As per SEER documentation (Tumor Size Over Time Recode - SEER Recodes, 2018), tumor size should range from 0–150 mm. This record will be removed to maintain data integrity, as imputation is not suitable for medical records.
Tumor Size	Certain values are missing.	Dropping the records with missing values.	Records with missing Tumor Size are dropped, as this critical clinical variable could bias cancer staging and mortality prediction if imputed.
Regional Node Examined	One value is missing.	Dropping the record with missing value.	Record with missing Regional Node Examined is dropped, as this critical clinical variable could bias cancer staging and mortality prediction if imputed.
Survival Months	Extreme outlier in Survival Months (760 which is approximately 63 years)	Dropping the record with extreme outlier	Compared to other records, 760 months is an extreme outlier (next highest is 107), so this record is dropped.
Mortality Status	Has inconsistent encoding (e.g., Alive, Dead, DEAD, dead, ALIVE, alive, ALive)	Capitalizing variable labels to ensure consistency across the dataset.	Capitalizing ensures uniform labels, preventing the model from misinterpreting identical categories as different.

Sex	Has inconsistent encoding (e.g., Female and 1).	Encoding "Female" as 0 to maintain consistency.	Nominal variables will be numerically encoded later; this standardizes the Sex variable for machine learning use.

Table 3 Data Cleaning Strategies for Breast Cancer Dataset

b) Implementation of suggested fixed with supporting evidence.

a. Removal of outliers in the Age variable

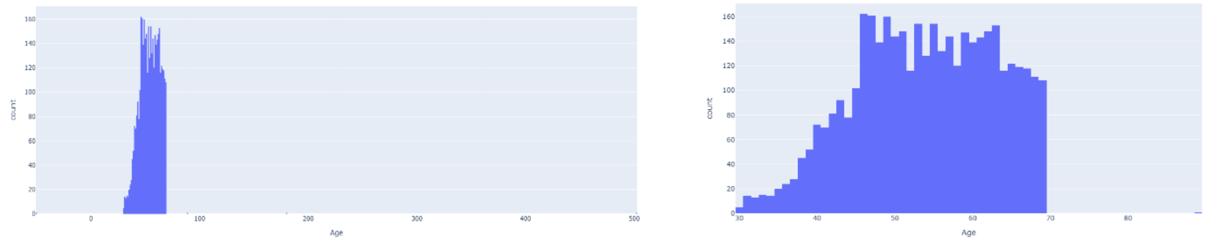
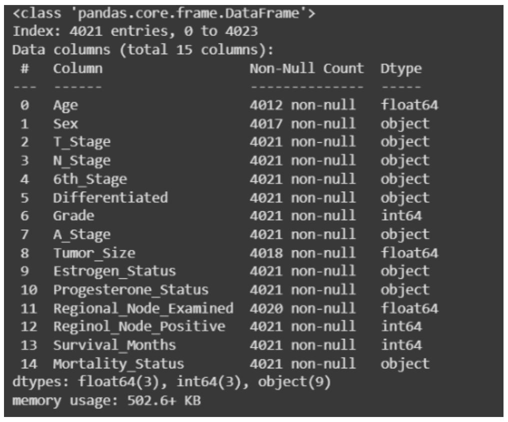


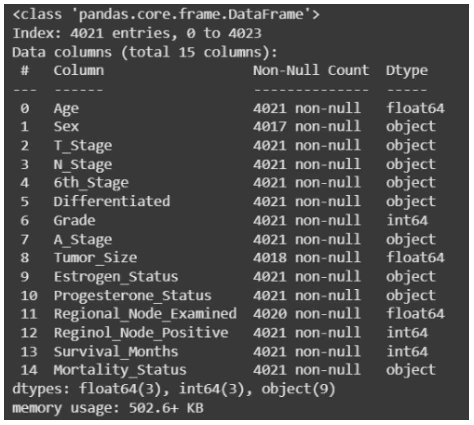
Figure 3 Before and After Removing Outliers in Age Variable

Although age 89 appears as an outlier, it is biologically plausible and clinically documented (BCRF, 2025), so it is retained as valid natural variability, unlike extreme errors (e.g., 500) that were removed.

b. Median imputation in the Age variable



```
<class 'pandas.core.frame.DataFrame'>
Index: 4021 entries, 0 to 4023
Data columns (total 15 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Age                     4012 non-null   float64
 1   Sex                     4017 non-null   object
 2   T_Stage                 4021 non-null   object
 3   N_Stage                 4021 non-null   object
 4   6th_Stage               4021 non-null   object
 5   Differentiated          4021 non-null   object
 6   Grade                   4021 non-null   int64
 7   A_Stage                 4021 non-null   object
 8   Tumor_Size              4018 non-null   float64
 9   Estrogen_Status         4021 non-null   object
 10  Progesterone_Status     4021 non-null   object
 11  Regional_Node_Examined  4020 non-null   float64
 12  Reginol_Node_Positive   4021 non-null   int64
 13  Survival_Months         4021 non-null   int64
 14  Mortality_Status        4021 non-null   object
dtypes: float64(3), int64(3), object(9)
memory usage: 502.6+ KB
```

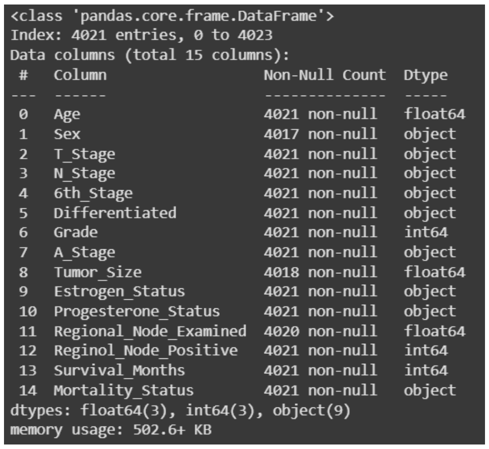


```
<class 'pandas.core.frame.DataFrame'>
Index: 4021 entries, 0 to 4023
Data columns (total 15 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Age                     4021 non-null   float64
 1   Sex                     4017 non-null   object
 2   T_Stage                 4021 non-null   object
 3   N_Stage                 4021 non-null   object
 4   6th_Stage               4021 non-null   object
 5   Differentiated          4021 non-null   object
 6   Grade                   4021 non-null   int64
 7   A_Stage                 4021 non-null   object
 8   Tumor_Size              4018 non-null   float64
 9   Estrogen_Status         4021 non-null   object
 10  Progesterone_Status     4021 non-null   object
 11  Regional_Node_Examined  4020 non-null   float64
 12  Reginol_Node_Positive   4021 non-null   int64
 13  Survival_Months         4021 non-null   int64
 14  Mortality_Status        4021 non-null   object
dtypes: float64(3), int64(3), object(9)
memory usage: 502.6+ KB
```

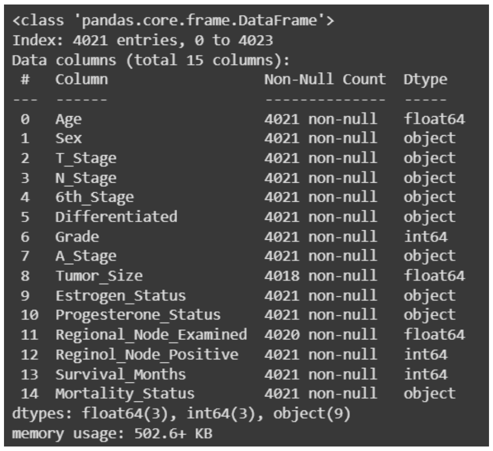
Figure 4 Before and After Imputing Median for Missing Values in Age

The non-null count for the Age variable increased to 4021 following median imputation.

c. Mode imputation in the Sex variable



```
<class 'pandas.core.frame.DataFrame'>
Index: 4021 entries, 0 to 4023
Data columns (total 15 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Age                     4021 non-null   float64
 1   Sex                     4017 non-null   object
 2   T_Stage                 4021 non-null   object
 3   N_Stage                 4021 non-null   object
 4   6th_Stage               4021 non-null   object
 5   Differentiated          4021 non-null   object
 6   Grade                   4021 non-null   int64
 7   A_Stage                 4021 non-null   object
 8   Tumor_Size              4018 non-null   float64
 9   Estrogen_Status         4021 non-null   object
 10  Progesterone_Status     4021 non-null   object
 11  Regional_Node_Examined  4020 non-null   float64
 12  Reginol_Node_Positive   4021 non-null   int64
 13  Survival_Months         4021 non-null   int64
 14  Mortality_Status        4021 non-null   object
dtypes: float64(3), int64(3), object(9)
memory usage: 502.6+ KB
```



```
<class 'pandas.core.frame.DataFrame'>
Index: 4021 entries, 0 to 4023
Data columns (total 15 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Age                     4021 non-null   float64
 1   Sex                     4021 non-null   object
 2   T_Stage                 4021 non-null   object
 3   N_Stage                 4021 non-null   object
 4   6th_Stage               4021 non-null   object
 5   Differentiated          4021 non-null   object
 6   Grade                   4021 non-null   int64
 7   A_Stage                 4021 non-null   object
 8   Tumor_Size              4018 non-null   float64
 9   Estrogen_Status         4021 non-null   object
 10  Progesterone_Status     4021 non-null   object
 11  Regional_Node_Examined  4020 non-null   float64
 12  Reginol_Node_Positive   4021 non-null   int64
 13  Survival_Months         4021 non-null   int64
 14  Mortality_Status        4021 non-null   object
dtypes: float64(3), int64(3), object(9)
memory usage: 502.6+ KB
```

Figure 5 Before and After Imputing Mode for Missing Values in Sex

The Sex field initially had inconsistent labels (“Female” and “1”); after encoding, the unique values are standardized as 0 (Female) and 1 (Male).

d. Removal of outliers in the Tumor Size variable

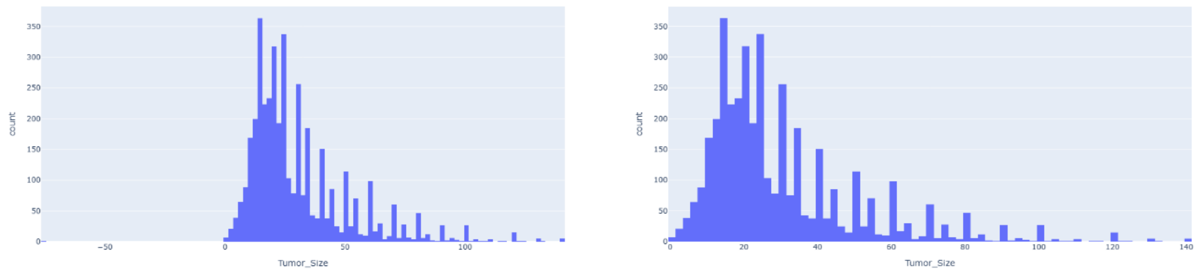
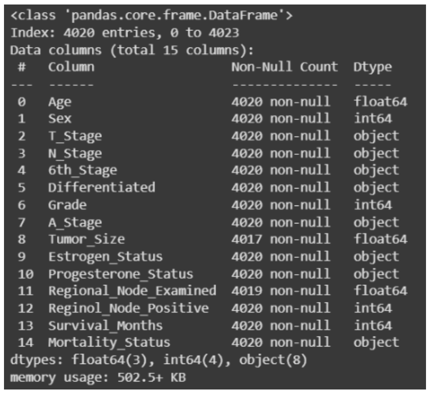


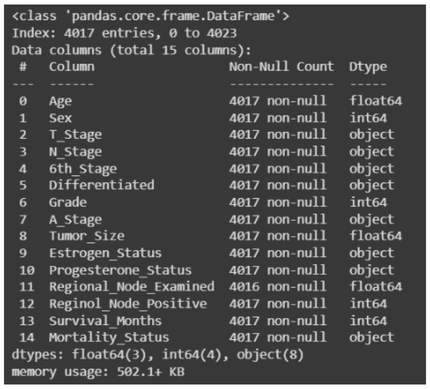
Figure 6 Before and After removing outliers in Tumor Size Variable

Although the updated histogram suggests statistical outliers, the maximum tumor size falls within the SEER-defined valid range of 0–150 mm and is medically acceptable. (Tumor Size Over Time Recode - SEER Recodes, 2018)

e. Dropping the records with missing Tumor Size values



```
<class 'pandas.core.frame.DataFrame'>
Index: 4020 entries, 0 to 4023
Data columns (total 15 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Age                     4020 non-null   float64
 1   Sex                     4020 non-null   int64
 2   T_Stage                 4020 non-null   object
 3   N_Stage                 4020 non-null   object
 4   6th_Stage               4020 non-null   object
 5   Differentiated          4020 non-null   object
 6   Grade                   4020 non-null   int64
 7   A_Stage                 4020 non-null   object
 8   Tumor_Size              4017 non-null   float64
 9   Estrogen_Status         4020 non-null   object
 10  Progesterone_Status     4020 non-null   object
 11  Regional_Node_Examined  4019 non-null   float64
 12  Reginol_Node_Positive   4020 non-null   int64
 13  Survival_Months         4020 non-null   int64
 14  Mortality_Status        4020 non-null   object
dtypes: float64(3), int64(4), object(8)
memory usage: 502.5+ KB
```

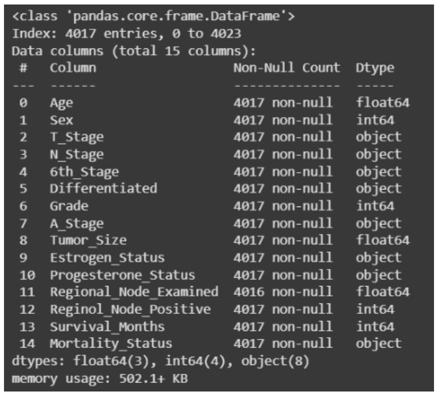


```
<class 'pandas.core.frame.DataFrame'>
Index: 4017 entries, 0 to 4023
Data columns (total 15 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Age                     4017 non-null   float64
 1   Sex                     4017 non-null   int64
 2   T_Stage                 4017 non-null   object
 3   N_Stage                 4017 non-null   object
 4   6th_Stage               4017 non-null   object
 5   Differentiated          4017 non-null   object
 6   Grade                   4017 non-null   int64
 7   A_Stage                 4017 non-null   object
 8   Tumor_Size              4017 non-null   float64
 9   Estrogen_Status         4017 non-null   object
 10  Progesterone_Status     4017 non-null   object
 11  Regional_Node_Examined  4016 non-null   float64
 12  Reginol_Node_Positive   4017 non-null   int64
 13  Survival_Months         4017 non-null   int64
 14  Mortality_Status        4017 non-null   object
dtypes: float64(3), int64(4), object(8)
memory usage: 502.1+ KB
```

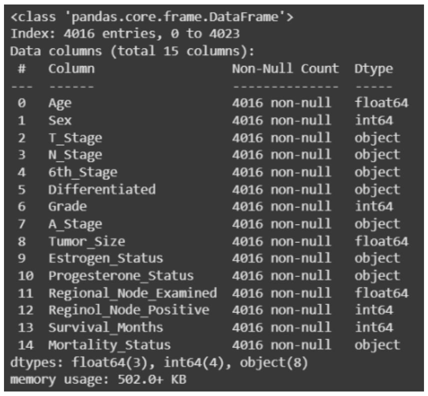
Figure 7 Before and After removing records with missing Tumor Size

The non-null count for other variables decreased to 4017, matching the Tumor Size variable, after dropping records with missing tumor size values.

f. Dropping the record with missing Regional Node Examined value



```
<class 'pandas.core.frame.DataFrame'>
Index: 4017 entries, 0 to 4023
Data columns (total 15 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Age                    4017 non-null   float64
 1   Sex                    4017 non-null   int64  
 2   T_Stage                4017 non-null   object 
 3   N_Stage                4017 non-null   object 
 4   6th_Stage              4017 non-null   object 
 5   Differentiated         4017 non-null   object 
 6   Grade                  4017 non-null   int64  
 7   A_Stage                4017 non-null   object 
 8   Tumor_Size             4017 non-null   float64
 9   Estrogen_Status        4017 non-null   object 
 10  Progesterone_Status    4017 non-null   object 
 11  Regional_Node_Examined 4016 non-null   float64
 12  Reginol_Node_Positive  4017 non-null   int64  
 13  Survival_Months        4017 non-null   int64  
 14  Mortality_Status       4017 non-null   object 
dtypes: float64(3), int64(4), object(8)
memory usage: 502.1+ KB
```

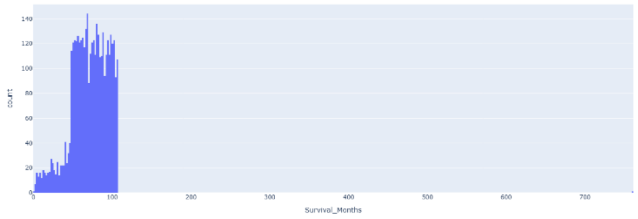


```
<class 'pandas.core.frame.DataFrame'>
Index: 4016 entries, 0 to 4023
Data columns (total 15 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Age                    4016 non-null   float64
 1   Sex                    4016 non-null   int64  
 2   T_Stage                4016 non-null   object 
 3   N_Stage                4016 non-null   object 
 4   6th_Stage              4016 non-null   object 
 5   Differentiated         4016 non-null   object 
 6   Grade                  4016 non-null   int64  
 7   A_Stage                4016 non-null   object 
 8   Tumor_Size             4016 non-null   float64
 9   Estrogen_Status        4016 non-null   object 
 10  Progesterone_Status    4016 non-null   object 
 11  Regional_Node_Examined 4016 non-null   float64
 12  Reginol_Node_Positive  4016 non-null   int64  
 13  Survival_Months        4016 non-null   int64  
 14  Mortality_Status       4016 non-null   object 
dtypes: float64(3), int64(4), object(8)
memory usage: 502.0+ KB
```

Figure 8 Before and After removing records with missing Regional Node Examined

The non-null count for other variables decreased to 4016, matching the Regional Node Examined variable, after dropping records with missing regional node examined values.

g. Removal of outliers in the Survival Months variable



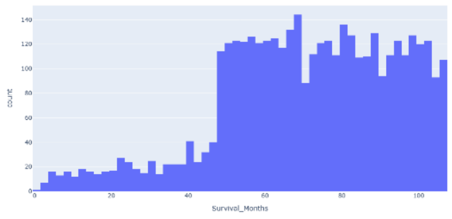
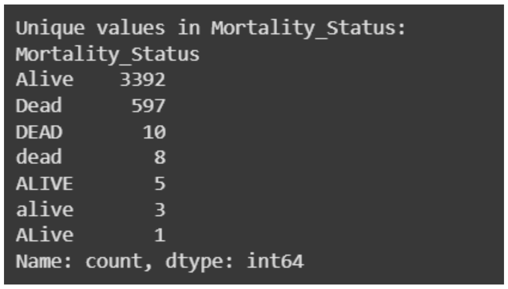


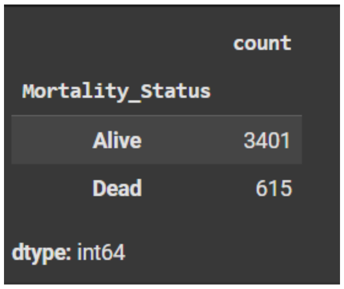
Figure 9 Before and After Removing Outliers in Survival Months Variable

The updated histogram confirms successful outlier removal from the data frame.

h. Capitalizing Mortality Status labels



```
Unique values in Mortality_Status:
Mortality_Status
Alive     3392
Dead       597
DEAD        10
dead         8
ALIVE        5
alive        3
ALive        1
Name: count, dtype: int64
```



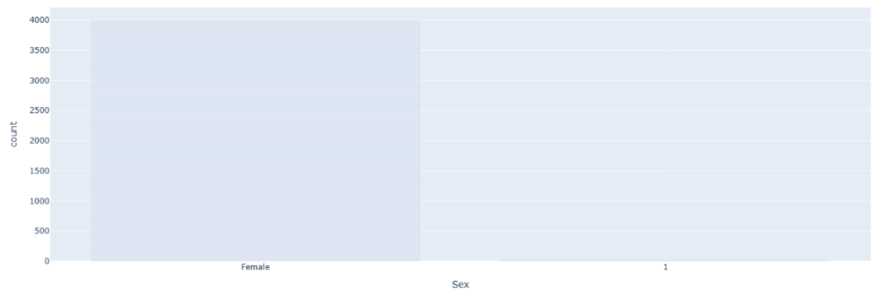
	count
Mortality_Status	
Alive	3401
Dead	615

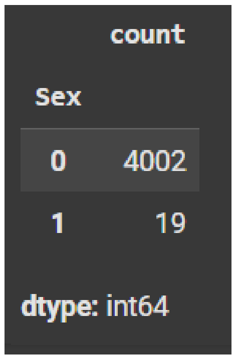
dtype: int64

Figure 10 Before and After Capitalizing Mortality Status Variable

The unique values of Mortality Status now confirm label consistency after capitalization.

i. Encoding inconsistent labels in the Sex variable





```
     count
Sex
0     4002
1       19

dtype: int64
```

Figure 11 Labels of Sex Variable: Before and After Encoding Sex Variable

The Sex field initially had inconsistent labels (“Female” and “1”); after encoding, the unique values are standardized as 0 (Female) and 1 (Male).

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

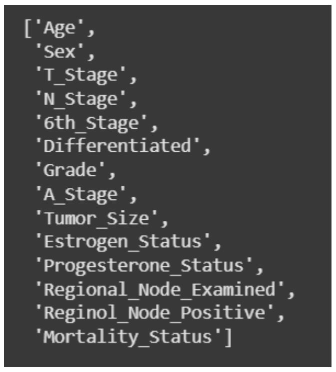
a) Clarification on each model’s type, learnable parameters, and key hyperparameters.

Algorithm Name	Algorithm Type	Learnable Parameters	Some Strategic Hyperparameters
NB	Parametric	Mean and variance of features for each class	- var_smoothing
LR	Parametric	Coefficients (weights) for each feature	- C (inverse regularization) - max_iter
KNN (K=9)	Non-parametric	None	- n_neighbors (K) - metric (distance metric) - weights

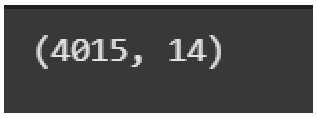
Table 4 Classification Algorithms for Breast Cancer Mortality Status Prediction

b) Training classification models

i. List of features and data dimensions



```
['Age',
 'Sex',
 'T_Stage',
 'N_Stage',
 '6th_Stage',
 'Differentiated',
 'Grade',
 'A_Stage',
 'Tumor_Size',
 'Estrogen_Status',
 'Progesterone_Status',
 'Regional_Node_Examined',
 'Reginol_Node_Positive',
 'Mortality_Status']
```



```
(4015, 14)
```

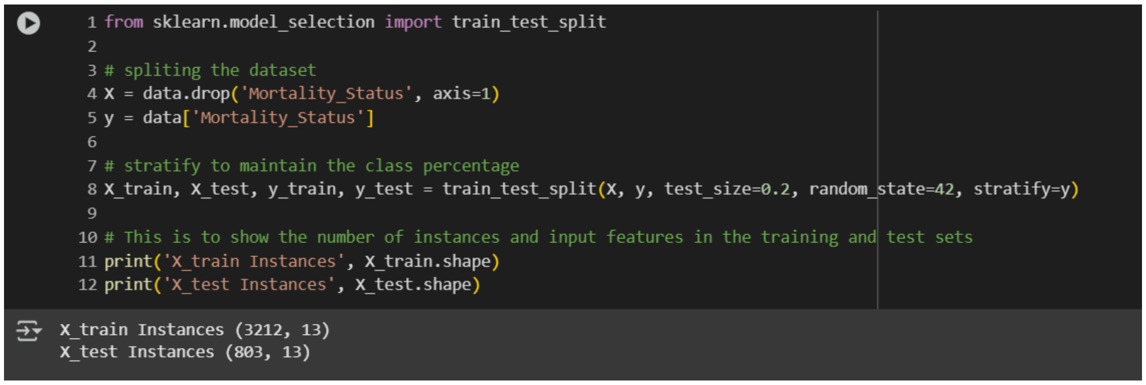
Figure 12 List of features and data dimensions

ii. Train-Test Split Considerations

When determining the train-test split for model evaluation, several critical factors must be considered. The dataset contains 4,015 records, which is moderately sized. As recommended by Kuhn and Johnson (2023) in *Applied Predictive Modeling*, an 80/20 split is appropriate for medium datasets, balancing performance estimation with adequate training data. Importantly, the dataset is heavily imbalanced, with 85% of instances labeled as ‘Alive’. Preserving this class distribution in both sets is essential to ensure valid evaluation metrics, particularly for the minority class (Dead). A smaller test set (e.g., 10%) may fail to include sufficient minority samples, compromising metrics like recall (Chawla et al., 2002). Conversely, a 70/30 split increases test reliability but may limit training effectiveness, especially for models like KNN and Logistic Regression. The 80/20 split offers a favorable bias-variance tradeoff and can be effectively combined with oversampling techniques such as SMOTE to improve minority-class prediction (Goodfellow, Bengio & Courville, 2016).

iii. Ensuring Consistency and Fairness in Model Evaluation

- Evidence to confirm that all models were evaluated using the same test split

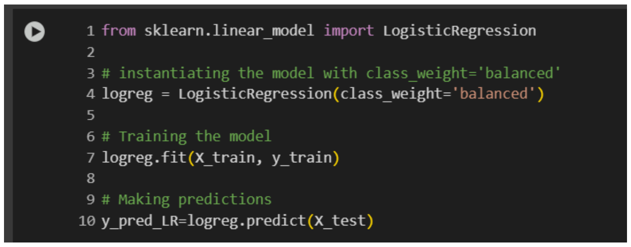


```
from sklearn.model_selection import train_test_split

# spliting the dataset
X = data.drop('Mortality_Status', axis=1)
y = data['Mortality_Status']

# stratify to maintain the class percentage
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# This is to show the number of instances and input features in the training and test sets
print('X_train Instances', X_train.shape)
print('X_test Instances', X_test.shape)
```

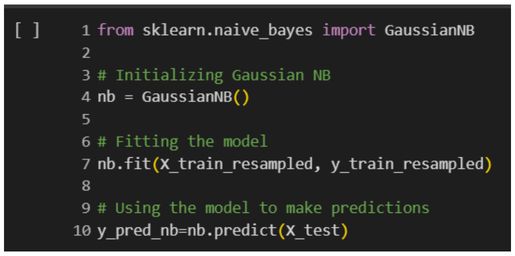
```
X_train Instances (3212, 13)
X_test Instances (803, 13)
```

Figure 13 Train-Test Split for Breast Cancer Mortality Status Prediction

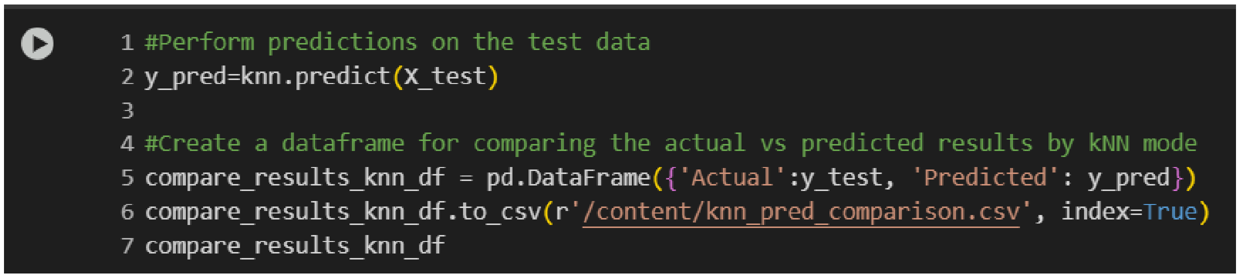
As shown above, the train-test split results in a test set with dimensions of (803, 13). The use of a fixed **random_state** ensures reproducibility, meaning all models are evaluated on the exact same test instances



```
from sklearn.linear_model import LogisticRegression

# instantiating the model with class_weight='balanced'
logreg = LogisticRegression(class_weight='balanced')

# Training the model
logreg.fit(X_train, y_train)

# Making predictions
y_pred_LR=logreg.predict(X_test)
```



```
from sklearn.naive_bayes import GaussianNB

# Initializing Gaussian NB
nb = GaussianNB()

# Fitting the model
nb.fit(X_train_resampled, y_train_resampled)

# Using the model to make predictions
y_pred_nb=nb.predict(X_test)
```



```
#Perform predictions on the test data
y_pred=knn.predict(X_test)

#Create a dataframe for comparing the actual vs predicted results by kNN mode
compare_results_knn_df = pd.DataFrame({'Actual':y_test, 'Predicted': y_pred})
compare_results_knn_df.to_csv(r'/content/knn_pred_comparison.csv', index=True)
compare_results_knn_df
```

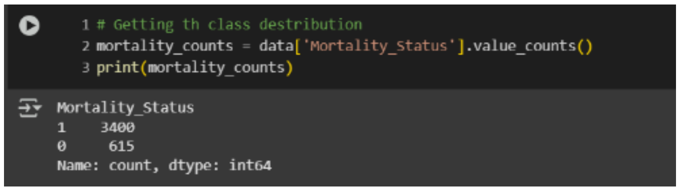
Figure 14 Model making predictions on X_test

The images illustrate that the initial base models, prior to hyperparameter tuning, are making predictions on X_test, which was derived using the train_test_split function as shown in the provided code snippet.

The confusion matrices for all models are provided in Task 5 (see Images 16, 17, and 18). The sum of TP, TN, FP, and FN equals 803 in each case, confirming that all models were evaluated on the same test set, as ensured by the fixed random state during the data split.

- Consistent Evidence Ratio of Mortality Status (“Alive” to “Dead”) in Training and Test Sets

As shown in the train-test split code (see Image 13), the parameter stratify=y has been set. This ensures that the class distribution of the target variable remains consistent between the training and test sets.



```
# Getting th class destribution
mortality_counts = data['Mortality_Status'].value_counts()
print(mortality_counts)
```

```
Mortality_Status
1    3400
0     615
Name: count, dtype: int64
```

Figure 15 Class Distribution on Mortality Status

It is evident that approximately 85% of the dataset consists of “Alive” records. This is supported by the confusion matrices provided in Task 5 (see Images 16, 17, and 18). Out of the 803 test records, 123 are labeled as “Dead” (approximately 15%), and 680 are labeled as “Alive” (approximately 85%). This class distribution confirms that stratification during the train-test split was successful.

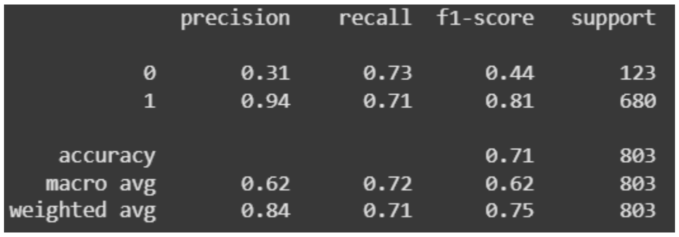
- Ensuring Reliable Model Evaluation with Stratified Sampling and Reproducibility

In Python Notebook 2, the line of code X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y), ensures two critical conditions essential for reliable model evaluation. Firstly, setting random_state=42 guarantees reproducibility by ensuring the same test set is used across all model evaluations, as supported by Raste et al. (2022). This consistency is crucial for fair performance comparison. Secondly, the use of stratify=y preserves the original class distribution of the target variable ("Alive" vs. "Dead" in Mortality Status) across both training and test subsets. For imbalanced datasets, such as this one where “Dead” cases form a minority, stratified sampling prevents class underrepresentation, reducing the risk of biased training and misleading evaluation (Sadaiyandi et al., 2023; Khan, 2022). By maintaining proportional representation of both classes, this approach enhances the validity, fairness, and comparability of model results, making the mortality prediction outcomes more trustworthy and clinically meaningful.

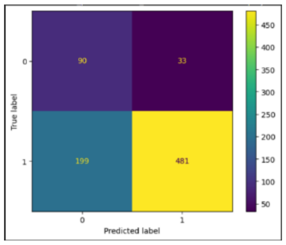
Task (5) – Evaluating your Cancer Mortality Status Classification Models

a) Evaluation metrics for each of the models

a. Logistic Regression



```
              precision    recall  f1-score   support

           0       0.31      0.73      0.44       123
           1       0.94      0.71      0.81       680

    accuracy                           0.71       803
   macro avg       0.62      0.72      0.62       803
weighted avg       0.84      0.71      0.75       803
```



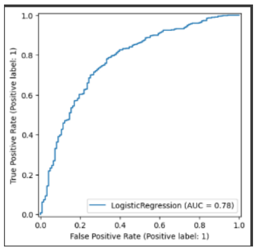
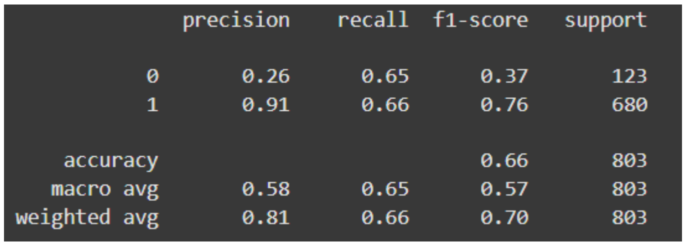
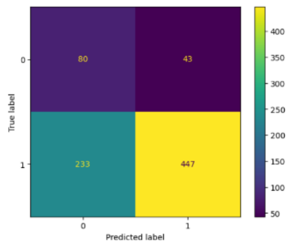


Figure 16 Evaluation metrics on Logistic Regression

b. K- Nearest Neighbors



```
              precision    recall  f1-score   support

           0       0.26      0.65      0.37       123
           1       0.91      0.66      0.76       680

    accuracy                           0.66       803
   macro avg       0.58      0.65      0.57       803
weighted avg       0.81      0.66      0.70       803
```



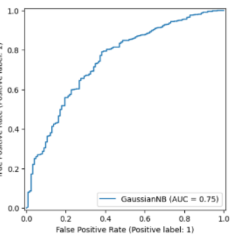
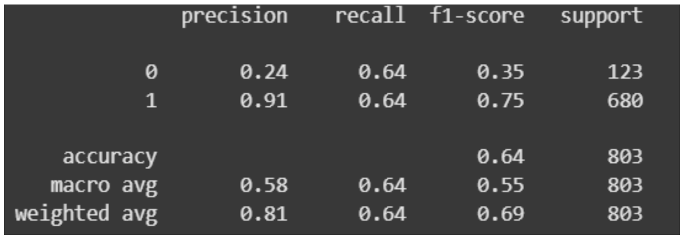
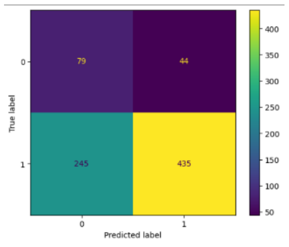


Figure 17 Evaluation metrics on KNN

c. Naive Bayes



```
              precision    recall  f1-score   support

           0       0.24      0.64      0.35       123
           1       0.91      0.64      0.75       680

    accuracy                           0.64       803
   macro avg       0.58      0.64      0.55       803
weighted avg       0.81      0.64      0.69       803
```



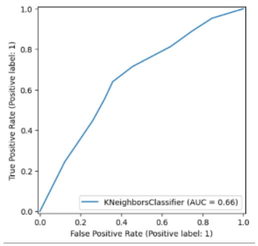


Figure 18 Evaluation metrics on Naive Bayes

b) Discussion on evaluation metrics

Metrics	**USE or DO NOT USE**	**Justification for choosing "USE" or "DO NOT USE" in relation to the success criteria**	**Model Name**	**Test Score**
Accuracy	DO NOT USE	With 85% "Alive" records, accuracy is misleading as it doesn't effectively measure class discrimination. A model could achieve high accuracy by predicting the majority class without any real discriminative ability.	*NB*	66%
			LR	71%
			KNN (K=9)	64%
Recall	USE	Recall measures the ability to correctly identify "Dead" patients, crucial for discriminating between "Dead" and "Alive" cases. In a medical setting, accurate classification of "Dead" patients is critical for saving lives.	*NB*	65%
			LR	73%
			KNN (K=9)	64%
Precision	USE	Precision measures how many predicted "Dead" patients were actually deceased, ensuring the model doesn't over-predict the "Dead" class and maintains its ability to discriminate between classes.	*NB*	26%
			LR	31%
			KNN (K=9)	24%
F-Score	USE	F-Score balances precision and recall, accounting for both false positives and negatives, aligning with success criteria that allow some misclassifications while ensuring good discrimination.	*NB*	37%
			LR	44%
			KNN (K=9)	35%
AUC-ROC	DO NOT USE	AUC-ROC can be misleading with significant class imbalance (85% "Alive" records), as small changes in false positives have minimal impact in datasets with few positive cases ("Dead" class), giving an overly optimistic view of performance.	*NB*	75%
			LR	78%
			KNN (K=9)	66%

Table 5 Evaluation Metric Analysis for Breast Cancer Mortality Prediction

Given the class imbalance with 'Dead' as the minority class, and its critical importance, the analysis focuses on recall, precision, and F-score for the 'Dead' class, using class-specific metrics over macro or weighted averages for better reflection of model performance.

c) Best model selection

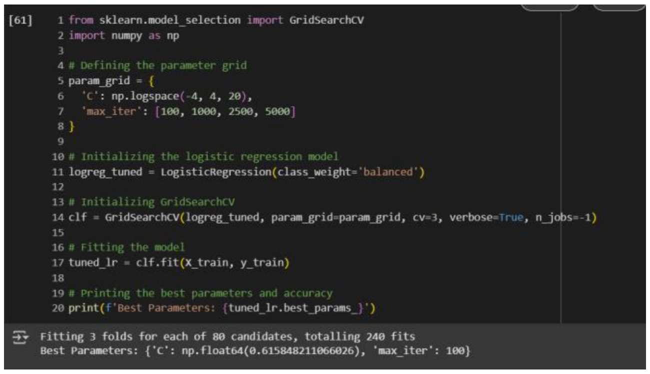
Based on the evaluation metrics, Logistic Regression is the best model for classifying mortality status. It outperforms Naive Bayes and KNN in F1-score, Precision, and Recall for the 'Dead' class (class 0), which directly aligns with the success criteria of better distinguishing between deceased and surviving cancer patients. Its superior F1-score reflects a strong balance between identifying true positives and minimizing false positives. While metrics like Accuracy and AUC-ROC may be less reliable for imbalanced datasets, Logistic Regression also leads in these, reinforcing its suitability as the optimal model for this classification task.

d) Hyperparameter tuning on Logistic Regression

i. Evidence of GridSearchCV implementation

The tuned parameters include **C** and **max_iter**, and notably, **class_weight='balanced'** was applied to address class imbalance. This approach proved more effective than using SMOTE, as the base model with class weights outperformed the model trained on SMOTE-resampled data.

GridSearchCV identified **C ≈ 0.616** and **max_iter = 100** as the optimal hyperparameters. The chosen C value balances underfitting and overfitting, while 100 iterations were sufficient for convergence, ensuring efficient training without high computational cost.



```
[61]
from sklearn.model_selection import GridSearchCV
import numpy as np

# Defining the parameter grid
param_grid = {
  'C': np.logspace(-4, 4, 20),
  'max_iter': [100, 1000, 2500, 5000]
}

# Initializing the logistic regression model
logreg_tuned = LogisticRegression(class_weight='balanced')

# Initializing GridSearchCV
clf = GridSearchCV(logreg_tuned, param_grid=param_grid, cv=3, verbose=True, n_jobs=-1)

# Fitting the model
tuned_lr = clf.fit(X_train, y_train)

# Printing the best parameters and accuracy
print(f'Best Parameters: {tuned_lr.best_params_}')
```

```
Fitting 3 folds for each of 80 candidates, totalling 240 fits
Best Parameters: {'C': np.float64(0.615848211066026), 'max_iter': 100}
```

Figure 19 GridSearchCV implementation on LR

ii. Performance analysis

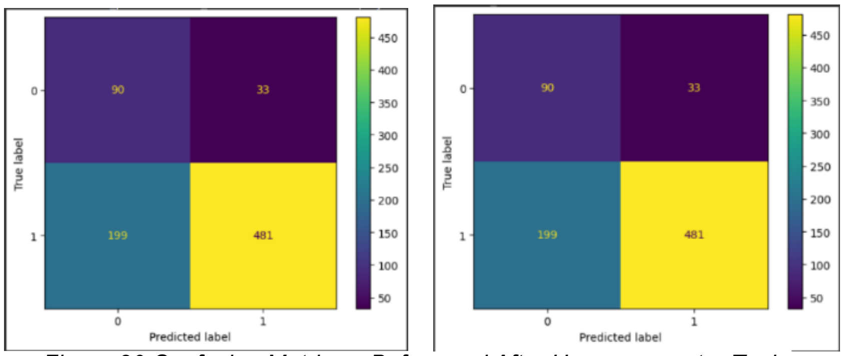
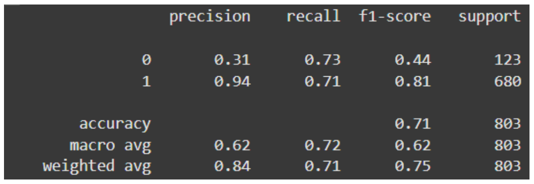


Figure 20 Confusion Matrices: Before and After Hyperparameter Tuning

The confusion matrix comparison shows that hyperparameter did not change model performance. Metrics remained identical, indicating the initial model was already well-suited for classifying "Dead" and "Alive" cancer patients, and further tuning offered no improvement.



```
              precision    recall  f1-score   support

           0       0.31      0.73      0.44       123
           1       0.94      0.71      0.81       680

    accuracy                           0.71       803
   macro avg       0.62      0.72      0.62       803
weighted avg       0.84      0.71      0.75       803
```

Figure 21 Classification Report: Hyperparameter tuned LR

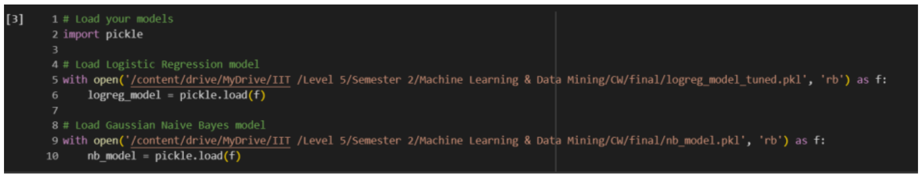
e) Critical analysis of the best model

- Limitations
  - With 232 misclassifications out of 803 patients, the model has an error rate of ~29%, which may be too high for critical medical decisions like mortality prediction.
  - The model still misclassifies many in the minority class, highlighting limitations in handling the dataset's class imbalance.
- Ethical considerations

- Misclassifying 33 deceased patients risks overly optimistic prognoses, affecting ethical care and end-of-life decisions.
- Misclassifying 199 survivors as deceased could lead to unnecessary treatments, raising ethical concerns about resource misallocation.
- The model may introduce bias in mortality predictions for underrepresented patient populations, affecting its accuracy across diverse demographic groups.

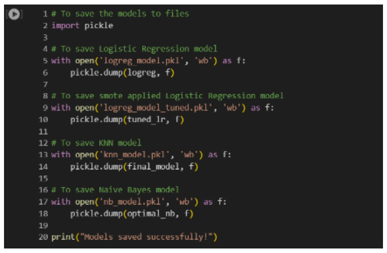
f) Probability based voting ensemble classifier

i. Python code blocks to import, declare base learners and fit ensemble learner



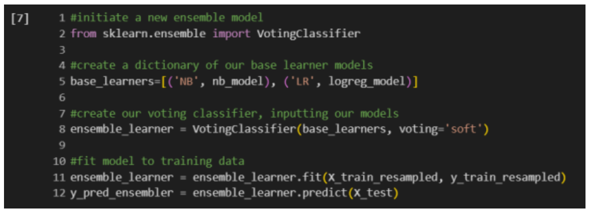
```
[3]
# Load your models
import pickle

# Load Logistic Regression model
with open('/content/drive/MyDrive/IIT /Level 5/Semester 2/Machine Learning & Data Mining/CW/final/logreg_model_tuned.pkl', 'rb') as f:
    logreg_model = pickle.load(f)

# Load Gaussian Naive Bayes model
with open('/content/drive/MyDrive/IIT /Level 5/Semester 2/Machine Learning & Data Mining/CW/final/nb_model.pkl', 'rb') as f:
    nb_model = pickle.load(f)
```

Figure 22 Code block to Import and Declare Base learners on Notebook 3



```
# To save the models to files
import pickle

# To save Logistic Regression model
with open('logreg_model.pkl', 'wb') as f:
    pickle.dump(logreg, f)

# To save smote applied Logistic Regression model
with open('logreg_model_tuned.pkl', 'wb') as f:
    pickle.dump(tuned_lr, f)

# To save KNN model
with open('knn_model.pkl', 'wb') as f:
    pickle.dump(final_model, f)

# To save Naive Bayes model
with open('nb_model.pkl', 'wb') as f:
    pickle.dump(optimal_nb, f)

print("Models saved successfully!")
```

Figure 24 Code Block to Save Models to Colab Environment in Notebook 2

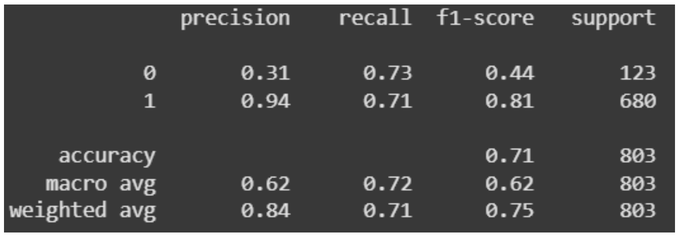


```
[7]
#initiate a new ensemble model
from sklearn.ensemble import VotingClassifier

#create a dictionary of our base learner models
base_learners=[('NB', nb_model), ('LR', logreg_model)]

#create our voting classifier, inputting our models
ensemble_learner = VotingClassifier(base_learners, voting='soft')

#fit model to training data
ensemble_learner = ensemble_learner.fit(X_train_resampled, y_train_resampled)
y_pred_ensembler = ensemble_learner.predict(X_test)
```

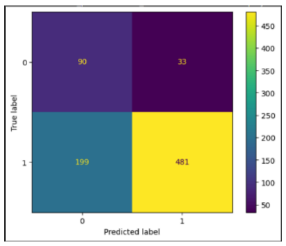
Figure 23 Fitting Ensemble Learner

ii. Comparison and Evaluation of Base Learners vs Ensemble Learner

- Evaluation metrics of the base learners



```
              precision    recall  f1-score   support

           0       0.31      0.73      0.44       123
           1       0.94      0.71      0.81       680

    accuracy                           0.71       803
   macro avg       0.62      0.72      0.62       803
weighted avg       0.84      0.71      0.75       803
```



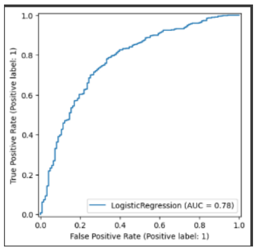
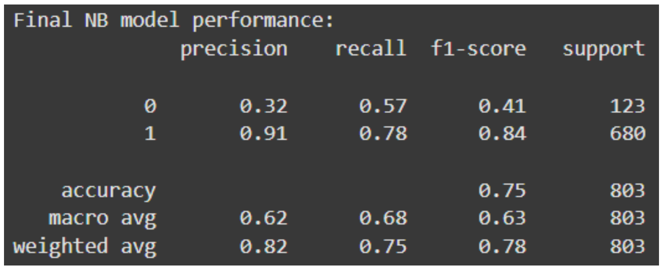
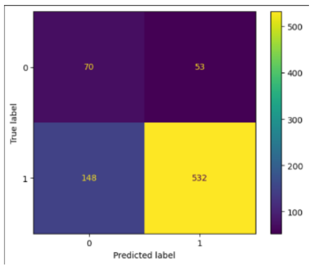


Figure 25 Evaluation metrics of base learner 1 | Logistic Regression



```
Final NB model performance:
              precision    recall  f1-score   support

           0       0.32      0.57      0.41       123
           1       0.91      0.78      0.84       680

    accuracy                           0.75       803
   macro avg       0.62      0.68      0.63       803
weighted avg       0.82      0.75      0.78       803
```



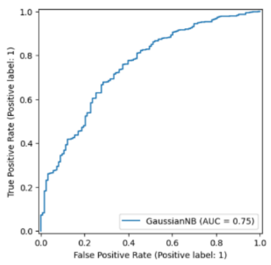
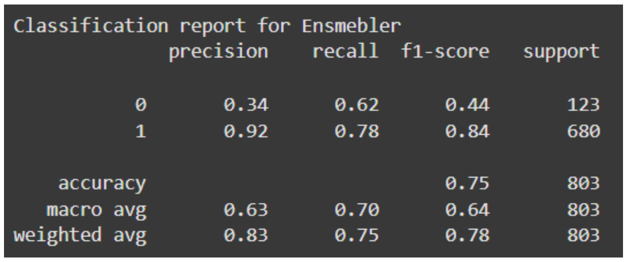
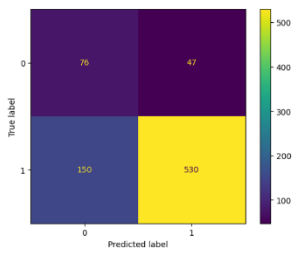


Figure 26 Evaluation metrics of base learner 2 | Naïve Bayes

- Evaluation of the ensemble learner



```
Classification report for Ensmebler
              precision    recall  f1-score   support

           0       0.34      0.62      0.44       123
           1       0.92      0.78      0.84       680

    accuracy                           0.75       803
   macro avg       0.63      0.70      0.64       803
weighted avg       0.83      0.75      0.78       803
```



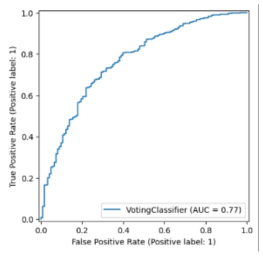


Figure 27 Evaluation metrics of Ensemble Learner

- Justification for the selection of the base learners

Logistic Regression (LR) showed the highest overall performance and is selected as the first base learner. K-Nearest Neighbors (KNN) and Naïve Bayes (NB), both tuned for optimal performance, had identical recall, but NB demonstrated better precision for the "Dead" class. While LR already achieves the best overall recall, NB adds value by excelling in classifying "Alive" instances. Together, they are likely to complement each other well in the ensemble model.

iii. Analysis on Ensemble Classifier

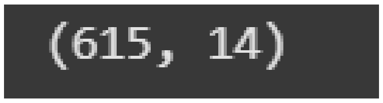
An analysis of the confusion matrices for the two base learners and the Ensemble Voting Classifier reveals key performance patterns. The ensemble displays intermediate performance between its base learners. Compared to Naïve Bayes (NB), it slightly improves in identifying deceased patients (76 vs. 70), but it identifies fewer alive patients (530 vs 532). Against Logistic Regression (LR), it identifies fewer deceased patients (76 vs. 90) but more alive patients correctly (534 vs. 481).

Given the healthcare context and the goal of accurately distinguishing between 'Dead' and 'Alive' cancer patients, correctly identifying those who will not survive is critically important**. Logistic Regression (LR)** is therefore recommended over the Ensemble Voting Classifier, as it identifies 90 deceased patients compared to only 76 by the Ensemble. Although LR produces more false positives (199 alive patients misclassified as deceased), this conservative approach is clinically preferred over missing terminal cases. From both ethical and medical perspectives, false negatives are more harmful than false positives. While the Ensemble shows modest improvement over Naïve Bayes, it does not exceed LR's effectiveness in identifying deceased patients, making LR more aligned with the priorities of healthcare professionals.

Case Study (B) Analyses Report for Predicting Survival Months Tasks

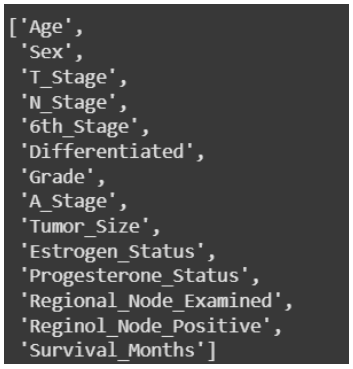
Task (1) – Domain Understanding and Designing Your Regression Experiments

The regression dataset contains 615 records because the task specifically targets predicting survival time for patients who did not survive cancer. To meet this objective, records of living patients were excluded, ensuring the model is trained only on deceased cases where survival duration is known and relevant.



```
(615, 14)
```

Figure 29 Data Dimensions



```
['Age',
 'Sex',
 'T_Stage',
 'N_Stage',
 '6th_Stage',
 'Differentiated',
 'Grade',
 'A_Stage',
 'Tumor_Size',
 'Estrogen_Status',
 'Progesterone_Status',
 'Regional_Node_Examined',
 'Reginol_Node_Positive',
 'Survival_Months']
```

Figure 28 List of Features

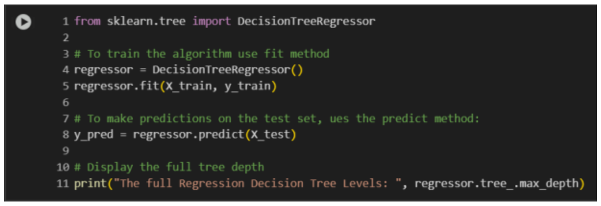
Task (2) – Modelling: Build Predictive Regression Models

a) Benefits of Using Decision Tree Regression for Breast Cancer Survival Prediction

Decision Tree (DT) regressors offer several advantages for predicting survival months in healthcare. They provide **interpretable survival predictions** through clear decision paths, showing how specific cancer characteristics (such as tumor size, node status, and hormone receptor status) impact survival time, making it easier for doctors to explain prognoses to patients. DTs also handle **mixed data types** naturally, accommodating both categorical variables (e.g., T Stage, N Stage, differentiation) and numerical ones (e.g., tumor size, age) without extensive preprocessing. Importantly, DTs can **capture non-linear survival patterns**, modeling complex interactions like the varying impact of hormone receptor status across cancer stages. They also offer **treatment prioritization insights** by highlighting which features most affect survival, guiding clinicians in tailoring therapies for patients with poor outcomes. Finally, DTs support **threshold-based decision making**, revealing critical values, such as specific tumor sizes or ages, where predicted survival changes sharply, aligning well with clinical decision-making processes.

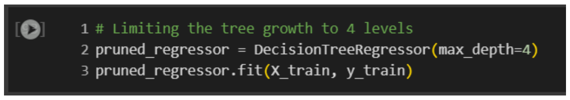
b) Building Decision Tree (DT) models

i. Python code blocks to import, declare and fit models



```python
from sklearn.tree import DecisionTreeRegressor

# To train the algorithm use fit method
regressor = DecisionTreeRegressor()
regressor.fit(X_train, y_train)

# To make predictions on the test set, ues the predict method:
y_pred = regressor.predict(X_test)

# Display the full tree depth
print("The full Regression Decision Tree Levels: ", regressor.tree_.max_depth)
```

Figure 31 Fitting Fully Grown DT



```python
# Limiting the tree growth to 4 levels
pruned_regressor = DecisionTreeRegressor(max_depth=4)
pruned_regressor.fit(X_train, y_train)
```

Figure 30 Fitting Pruned DT

ii. Impact of Pre-Pruning with Max Depth on DT-2

Pre-pruning a Decision Tree Regressor by setting the max_depth parameter to 4 is a constraint-based pruning technique that limits the tree's growth during construction. In the context of cancer survival month prediction, this approach offers several advantages. It helps **reduce overfitting** by preventing the model from capturing noise in the survival data, thus improving its **generalization** to new patient cases. The resulting model is also **more interpretable**, allowing healthcare professionals to better understand which features influence survival time. Additionally, it **lowers computational costs** by restricting tree complexity. However, there are notable disadvantages. Pre-pruning may **oversimplify complex relationships** between patient features and survival outcomes, potentially **missing important interactions** that occur at deeper levels of the tree. The uniform depth constraint could **cut off predictive paths** relevant to specific patient subgroups, leading to biased predictions. Furthermore, this simplification can **reduce predictive accuracy**, which is particularly concerning in clinical settings where even minor errors in estimating survival time can impact treatment decisions.

c) Graphical Representations of Decision Trees

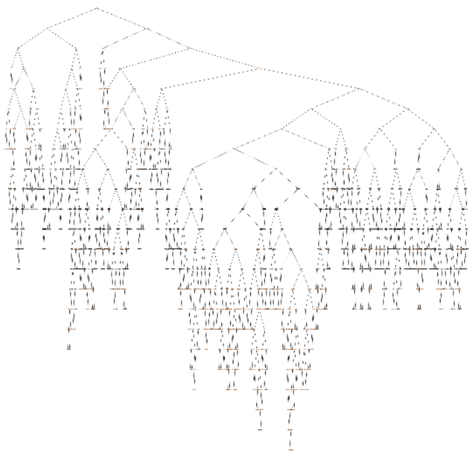


Figure 33 Fully Grown DT

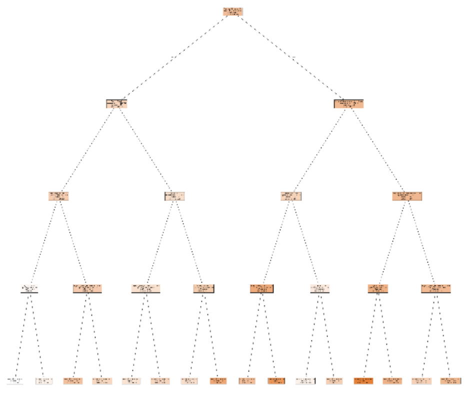


Figure 32 Pruned DT

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

a) Discussion on evaluation metrics

Metrics	USE or DO NOT USE	Justification in relation to the success criteria	Model Name	Test Score
MSE	USE	MSE highlights large prediction errors, making it useful for identifying models that minimize critical mistakes in survival estimates and aid treatment planning.	DT-1 (Fully Grown DT)	1258.1544715447155
			DT-2 (Pruned DT)	597.3107223280356
MAE	USE	MAE directly measures average errors in months, making it intuitive and aligned with the success criteria by highlighting even small survival prediction errors that matter in treatment planning.	DT-1 (Fully Grown DT)	29.471544715447155
			DT-2 (Pruned DT)	19.87145912176665
R-Square	DO NOT USE	R-squared is unsuitable as it shows variance explained, not actual errors in survival months. It lacks clinical relevance and can mask small but critical prediction errors, which are vital for treatment prioritization.	DT-1 (Fully Grown DT)	-1.3628940376913659
			DT-2 (Pruned DT)	-0.12178748822884655

Table 6 Evaluation Metric Analysis for Breast Cancer Survival Months Prediction

b) Best Model Selection

DT-2 (Pruned Decision Tree) is recommended as the best regression model for predicting cancer survival months, outperforming DT-1 in both key performance metrics. DT-2 has a lower MAE of 19.87 months compared to DT-1's 29.47 months, indicating more accurate predictions with an average error of 10 months less. Additionally, DT-2 demonstrates a significantly lower MSE of 597.31, compared to 1258.15 for DT-1, reflecting fewer large errors in its predictions. This pruned model better fulfills the success criteria by emphasizing the need for accurate survival month predictions, which are crucial for treatment prioritization. The lower MAE ensures more precise survival estimates, essential for life-saving decisions, while the reduced MSE highlights that DT-2 avoids potentially dangerous large prediction errors. Although both models show negative R-squared values, DT-2's superior performance in MAE and MSE confirms its reliability for accurate survival predictions, directly supporting the goal of improving treatment prioritization.

c) Concerns Regarding the Selected Performance Metrics

1. The metrics do not indicate if the model tends to overpredict or underpredict survival time, though this distinction is important for treatment planning.
2. The metrics may either downplay or exaggerate errors in patients with complex conditions, affecting the model's reliability for such cases.
3. While one model may appear more accurate than another, both may still produce errors large enough to impact care decisions.
4. The results do not show how certain the model is about its predictions, making it harder to judge their reliability in clinical use.

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

1. Root Node: Estrogen_Status <= 0.5?

   Patient's Estrogen Status = Negative (0)

   Decision: Yes -> Continue to the left subtree for further evaluation.

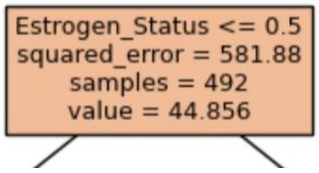


Figure 34 Root Node

2. Second Node: Age <= 0.056 (Scaled value)?

   Patient's Age = -2.14306158459098 (scaled value = (original value - mean) / standard deviation)

   Decision: Yes -> Continue to the left subtree for further evaluation.

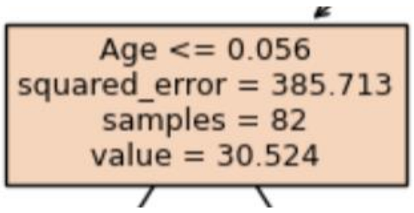


Figure 35 Second Node

3. Third Node: 6th_Stage <= -0.648?

   Patient's 6th_Stage = 1.414427157001414 (Scaled value)

   Decision: No -> Continue to the right subtree for further evaluation.

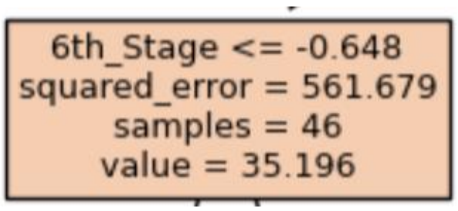


Figure 36 Third Node

4. Fourth Node: Regional_Node_Examined <= 0.263?

   Patient's Regional_Node_Examined = -1.153064848320758

   Decision: Yes -> Continue to the left subtree for further evaluation.

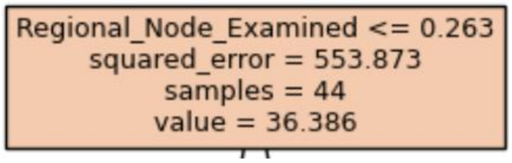


Figure 37 Fourth Node

5. Final Leaf Node: The node shows "value = 43.136" with 22 samples and squared error of 615.39

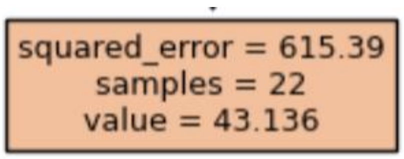


Figure 38 Final Leaf Node

The model predicts that patient B002565 has an estimated survival time of approximately 43.14 months (or around 3.6 years).

Reference

- BCRF. (2025). Breast Cancer in the Elderly | BCRF. *Breast Cancer Research Foundation*. Available from https://www.bcrf.org/about-breast-cancer/breast-cancer-elderly/.

- Chawla, N.V. et al. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16 (16), 321–357. Available from https://doi.org/10.1613/jair.953.

- GeeksforGeeks. (2020). String manipulations in Pandas DataFrame. *GeeksforGeeks*. Available from https://www.geeksforgeeks.org/string-manipulations-in-pandas-dataframe/.

- GeeksforGeeks. (2024). What is Data Leakage? *GeeksforGeeks*. Available from https://www.geeksforgeeks.org/what-is-data-leakage/.

- Goodfellow, I., Bengio, Y. and Courville, A. (2016). Deep Learning. *Deeplearningbook.org*. Available from https://www.deeplearningbook.org/.

- Gusarova, M. (2023). How to improve logistic regression in imbalanced data with class weights. *Medium*. Available from https://medium.com/@data.science.enthusiast/how-to-improve-logistic-regression-in-imbalanced-data-with-class-weights-1693719136aa.

- Illuri, L.T. (2023). Understanding Weighted k-Nearest Neighbors (k-NN) Algorithm. *Medium*. Available from https://medium.com/@lakshmiteja.ip/understanding-weighted-k-nearest-neighbors-k-nn-algorithm-3485001611ce.

- Khan, A.A. (2022). Balanced Split: A new train-test data splitting strategy for imbalanced datasets. *arXiv.org*. Available from https://arxiv.org/abs/2212.11116.

- Kuhn, M. and Johnson, K. (2023). *Applied Predictive Modeling*. Available from https://www.ic.unicamp.br/~wainer/cursos/1s2021/432/2013_Book_AppliedPredictiveModeling.pdf.

- machine learning - How to tune GaussianNB? (no date). *Stack Overflow*. Available from https://stackoverflow.com/questions/39828535/how-to-tune-gaussiannb.

- pandas.DataFrame.value_counts — pandas 1.4.4 documentation. (no date). *pandas.pydata.org*. Available from https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.value_counts.html.

- Qiao, F. (2019). Logistic Regression Model Tuning with scikit-learn — Part 1. *Medium*. Available from https://medium.com/data-science/logistic-regression-model-tuning-with-scikit-learn-part-1-425142e01af5.

- Raste, S. et al. (2022). Quantifying Inherent Randomness in Machine Learning Algorithms. *arXiv.org*. Available from https://doi.org/10.48550/arXiv.2206.12353.

- Sadaiyandi, J. et al. (2023). Stratified Sampling-Based Deep Learning Approach to Increase Prediction Accuracy of Unbalanced Dataset. *Electronics*, 12 (21), 4423. Available from https://doi.org/10.3390/electronics12214423.

- SMOTE for Imbalanced Classification with Python. (2024). *GeeksforGeeks*. Available from https://www.geeksforgeeks.org/smote-for-imbalanced-classification-with-python/.

- Subash A. (2023). Saving and Loading Trained Machine Learning Models with Python: A Comprehensive Guide. *Medium*. Available from https://medium.com/@subashdhoni86/saving-and-loading-trained-machine-learning-models-with-python-a-comprehensive-guide-b6e661b4ec01 [Accessed 4 May 2025].

- Tumor Size Over Time Recode - SEER Recodes. (2018). *SEER*. Available from https://seer.cancer.gov/tumorsize/.