

Analyses Report of Two Case Studies, Predicting Cancer Patients Mortality Status and Survival Months

Name: Senuri Hansamini Wedamulla

Table of Contents

Case Study (A): Predicting Cancer Patients Mortality Status.	3
Task (1) – Domain Understanding: Classification	3
Task (2) – Exploring and Understanding Your Dataset	3
2.a. Descriptive Statistics	3
2.c. Target Variable Distribution	4
Task (3) – Data Preparation: Cleaning and Transforming your data	5
3.a. Identify Issues and Propose Fixes	5
3.b. Screenshots of evidence (before and after)	5
4.a Algorithms	7
4.b. Build Predictive Classification Models	7
5.a. AUC-ROC curve graphs	8
5.b. Classification Evaluation Metrics	8
5.c. Best mortality status classification model	8
5.e. Critique of the Best-Performing Model and Ethical Considerations	10
5.f. Combining Base Learners	10
Case Study (B): Predicting Cancer Patients Survival Months.	13
Task (1) – Domain Understanding and Designing Your Regression Experiments	13
Task (2) – Modelling: Build Predictive Regression Models	14
2.a. Benefits of using a DT regressor	14
2.b. Build and Test Decision Tree (DT) regression models	14
2.c. Regression Decision Trees	15
Task (3) – Evaluating your Cancer Survival Months DT Regression Models	16
3.a. Regression Evaluation Metrics	16
3.b. Single Best Regression Model	16
3.c. Concerns On Selected Performance Metric/s	17
Task (4) – Interpreting Cancer Survival Months Decision Tree Outcomes	17

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

Task (1) – Domain Understanding: Classification

Variable Name	RETAIN or DROP	Brief justification for retention or dropping
Patient ID	DROP	Not relevant for prediction, serves only as a unique identifier.
Month of Birth	DROP	Not clinically relevant to mortality predictions.
Age	RETAIN	An important clinical factor, age impacts prognosis and treatment response.
Sex	DROP	Has only one valid unique value.
Occupation	DROP	Contains too many null or missing values
T Stage	RETAIN	Indicates tumor size/extent.
N Stage	RETAIN	Indicates lymph node involvement, key factor in prognosis.
6th Stage	RETAIN	A strong predictor of survival outcome.
Differentiated	RETAIN	Reflects how abnormal the tumor cells look
Grade	RETAIN	Describes tumor aggressiveness
A Stage	RETAIN	Indicates extent of spread
Tumour Size	RETAIN	Directly affects staging and survival probability.
Estrogen Status	RETAIN	Hormone receptor status, influences treatment and prognosis.
Progesterone Status	RETAIN	Complements estrogen status
Regional Node Examined	RETAIN	Indicates extent of diagnostic assessment
Regional Node Positive	RETAIN	Critical prognostic factor.
Survival Months	DROP	Used only during model training, Not needed as an input feature for prediction.
Mortality Status	RETAIN	Target variable for classification model; essential outcome measure.

Task (2) – Exploring and Understanding Your Dataset

To begin the analysis, basic steps were taken to understand the retained input features and the target variable, Mortality_Status.

2.a. Descriptive Statistics

Descriptive statistics were generated for all variables in the dataset using Python.

```
<class 'pandas.core.frame.DataFrame'>
Index: 4015 entries, 0 to 4023
Data columns (total 14 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Age                    4015 non-null   int64
1   T_Stage                4015 non-null   int64
2   N_Stage                4015 non-null   int64
3   6th_Stage              4015 non-null   int64
4   Differentiated         4015 non-null   int64
5   Grade                  4015 non-null   int64
6   A_Stage                4015 non-null   int64
7   Tumor_Size             4015 non-null   int64
8   Estrogen_Status        4015 non-null   int64
9   Progesterone_Status    4015 non-null   int64
10  Regional_Node_Examined  4015 non-null   int64
11  Regional_Node_Positive  4015 non-null   int64
12  Survival_Months         4015 non-null   int64
13  Mortality_Status        4015 non-null   int64
dtypes: int64(14)
memory usage: 470.5 KB
```

Figure 1: Data info

	Age	T_Stage	N_Stage	6th_Stage	Differentiated	Grade	A_Stage	Tumor_Size	Estrogen_Status	Progesterone_Status	Regional_Node_Examined	Reginol_Node_Positive	Survival_Months	Mortality_Status
count	4015.000000	4015.000000	4015.000000	4015.000000	4015.000000	4015.000000	4015.000000	4015.000000	4015.000000	4015.000000	4015.000000	4015.000000	4015.000000	4015.000000
mean	53.993524	1.783562	1.437111	2.319303	0.689166	2.151183	0.977335	30.362889	0.933001	0.826899	14.364882	4.147198	71.301121	0.846824
std	8.971774	0.764971	0.692668	1.265527	1.015338	0.637874	0.148852	20.896703	0.250051	0.378382	8.128184	5.094083	22.921993	0.360202
min	30.000000	1.000000	1.000000	1.000000	0.000000	1.000000	0.000000	1.000000	0.000000	0.000000	1.000000	1.000000	1.000000	0.000000
25%	47.000000	1.000000	1.000000	1.000000	0.000000	2.000000	1.000000	16.000000	1.000000	1.000000	9.000000	1.000000	56.000000	1.000000
50%	54.000000	2.000000	1.000000	2.000000	0.000000	2.000000	1.000000	25.000000	1.000000	1.000000	14.000000	2.000000	73.000000	1.000000
75%	61.000000	2.000000	2.000000	3.000000	1.000000	3.000000	1.000000	38.000000	1.000000	1.000000	19.000000	5.000000	90.000000	1.000000
max	89.000000	4.000000	3.000000	5.000000	3.000000	4.000000	1.000000	140.000000	1.000000	1.000000	61.000000	46.000000	107.000000	1.000000

Figure 2: Data describe

2.b. Variable Scale Types

Each variable in the dataset was examined and classified according to its measurement scale type. The table below shows the variable names and their corresponding types,

Variable Name	Scale Type
Age	Ratio
T_Stage	Ordinal
N_Stage	Ordinal
6 th _Stage	Ordinal
Differentiated	Nominal
Grade	Ordinal
A_Stage	Ordinal
Tumor_Size	Ratio
Estrogen_Status	Nominal
Progesterone_Status	Nominal
Regional_Node_Examined	Ratio
Reginol_Node_Positive	Ratio
Mortality_Status (Target)	Nominal (Binary)

This classification helped to identify which variables are suitable for categorical processing and which are numerical, which is essential for appropriate model selection and preprocessing.

2.c. Target Variable Distribution

The distribution of the target variable Mortality_Status was visualized using a bar chart. The plot showed the number of patients who are alive (labelled as 1) and those who died (labelled as 0). The plot showed that the number of patients in each class was not equal, causing an imbalance.

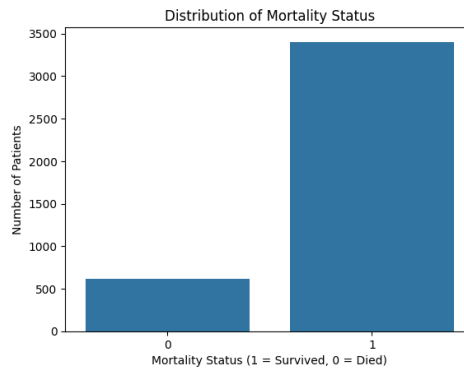


Figure 3: Distribution of Mortality Status

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

3.a. Identify Issues and Propose Fixes

Variable Name	Issue found	Proposed fix	Justification for used fix method
Age	Contained negative values and incorrect values, float values and few missing values	Identified that there's only two effected rows and dropped those two columns, converted the type to int and replaced the missing values with mean.	Ensures realistic age data and consistency in type for modeling.
Tumor_Size	Contained negative values and high outliers, float values and few missing values	Identified four effected rows and dropped them, converted the type to int and replaced the missing values with mean value	Removes unrealistic values and ensures consistency.
Survival_Months	Contained a high outlier	Identified and dropped the necessary row.	Removes unrealistic values.
Estrogen_Status	Contained String values , not numerically encoded and few missing values.	Mapped the string values 'Positive' and 'Negative' to 1 and 0, replaced the missing values with mean value	Allows numerical compatibility.
Mortality_Status	Contained spelling case variations	Standardize the values to “alive” and “dead” and mapped them as 1 and 0	Allows numerical compatibility.
Progesterone_Status	Contained String values , not numerically encoded.	Mapped the string values to respective numerical values.	Allows numerical compatibility.
T_Stage			
N_Stage			
6th_Stage			
Differentiated		Applied Label Encoder for ordinal encoding	Converts ordinal levels into numerical format.
A_Stage			
Regional_Node_Examined	Contained float values	Converted the type to int	Ensures consistency in type for modeling.

3.b. Screenshots of evidence (before and after)

1. Before

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4024 entries, 0 to 4023
Data columns (total 18 columns):
#  Column                Non-Null Count  Dtype
---  --
0  Patient_ID             4024 non-null  object
1  Month_of_Birth         4024 non-null  int64
2  Age                   4015 non-null  float64
3  Sex                   4020 non-null  object
4  Occupation            43 non-null    object
5  T_Stage               4024 non-null  object
6  N_Stage               4024 non-null  object
7  6th_Stage             4024 non-null  object
8  Differentiated         4024 non-null  object
9  Grade                 4024 non-null  int64
10 A_Stage              4024 non-null  object
11 Tumor_Size           4021 non-null  float64
12 Estrogen_Status      4024 non-null  object
13 Progesterone_Status  4024 non-null  object
14 Regional_Node_Examined 4021 non-null  float64
15 Regional_Node_Positive 4024 non-null  int64
16 Survival_Months      4024 non-null  int64
17 Mortality_Status     4024 non-null  object
dtypes: float64(3), int64(4), object(11)
memory usage: 566.0+ KB

```

Figure 4: Data info before

	0	Differentiated	0.000000
Patient_ID	0.000000	Grade	0.000000
Month_of_Birth	0.000000	A_Stage	0.000000
Age	0.223658	Tumor_Size	0.074553
Sex	0.099404	Estrogen_Status	0.000000
Occupation	98.931412	Progesterone_Status	0.000000
T_Stage	0.000000	Regional_Node_Examined	0.024851
N_Stage	0.000000	Reginol_Node_Positive	0.000000
6th_Stage	0.000000	Survival_Months	0.000000
		Mortality_Status	0.000000

Figure 5: Null values before

	Month_of_Birth	Age	Grade	Tumor_Size	Regional_Node_Examined	Reginol_Node_Positive	Survival_Months
count	4024.000000	4015.000000	4024.000000	4021.000000	4023.000000	4024.000000	4024.000000
mean	6.481362	54.107098	2.150596	30.419299	14.373602	4.158052	71.472167
std	3.475442	11.715528	0.638234	21.161080	8.129293	5.109331	25.361855
min	1.000000	-50.000000	1.000000	-75.000000	1.000000	1.000000	1.000000
25%	3.000000	47.000000	2.000000	16.000000	9.000000	1.000000	56.000000
50%	6.000000	54.000000	2.000000	25.000000	14.000000	2.000000	73.000000
75%	10.000000	61.000000	3.000000	38.000000	19.000000	5.000000	90.000000
max	12.000000	502.000000	4.000000	140.000000	61.000000	46.000000	760.000000

Figure 6: Data describe before

	Patient_ID	Sex	Occupation	T_Stage	N_Stage	6th_Stage	Differentiated	A_Stage	Estrogen_Status	Progesterone_Status	Mortality_Status
count	4024	4020	43	4024	4024	4024	4024	4024	4024	4024	4024
unique	4024	2	40	4	3	5	4	2	2	2	7
top	A4035	Female	House Person	T2	N1	IIA	Moderately differentiated	Regional	Positive	Positive	Alive
freq	1	4001	2	1786	2732	1305	2351	3932	3755	3326	3399

Figure 7: Data describe object before

2. After

```

<class 'pandas.core.frame.DataFrame'>
Index: 4015 entries, 0 to 4023
Data columns (total 14 columns):
#  Column                Non-Null Count  Dtype
---  --
0  Age                   4015 non-null  int64
1  T_Stage               4015 non-null  int64
2  N_Stage               4015 non-null  int64
3  6th_Stage             4015 non-null  int64
4  Differentiated         4015 non-null  int64
5  Grade                 4015 non-null  int64
6  A_Stage               4015 non-null  int64
7  Tumor_Size            4015 non-null  int64
8  Estrogen_Status       4015 non-null  int64
9  Progesterone_Status   4015 non-null  int64
10 Regional_Node_Examined 4015 non-null  int64
11 Regional_Node_Positive 4015 non-null  int64
12 Survival_Months      4015 non-null  int64
13 Mortality_Status     4015 non-null  int64
dtypes: int64(14)
memory usage: 478.5 KB

```

Figure 8: Data info after

	0	Tumor_Size	0.0
Age	0.0	Estrogen_Status	0.0
T_Stage	0.0	Progesterone_Status	0.0
N_Stage	0.0	Regional_Node_Examined	0.0
6th_Stage	0.0	Reginol_Node_Positive	0.0
Differentiated	0.0	Survival_Months	0.0
Grade	0.0	Mortality_Status	0.0
A_Stage	0.0		

Figure 9: Null values after

	Age	T_Stage	N_Stage	6th_Stage	Differentiated	Grade	A_Stage	Tumor_Size	Estrogen_Status	Progesterone_Status	Regional_Node_Examined	Reginol_Node_Positive	Survival_Months	Mortality_Status
count	4015.000000	4015.000000	4015.000000	4015.000000	4015.000000	4015.000000	4015.000000	4015.000000	4015.000000	4015.000000	4015.000000	4015.000000	4015.000000	4015.000000
mean	53.993524	1.783362	1.437111	2.315303	0.689166	2.151183	0.977335	30.362889	0.933001	0.826899	14.364882	4.147198	71.301121	0.846824
std	8.971774	0.764971	0.692668	1.265527	1.015338	0.637874	0.148852	20.896703	0.250051	0.378382	8.128184	5.094083	22.921993	0.360202
min	30.000000	1.000000	1.000000	1.000000	0.000000	1.000000	0.000000	1.000000	0.000000	0.000000	1.000000	1.000000	1.000000	0.000000
25%	47.000000	1.000000	1.000000	1.000000	0.000000	2.000000	1.000000	16.000000	1.000000	1.000000	9.000000	1.000000	56.000000	1.000000
50%	54.000000	2.000000	1.000000	2.000000	0.000000	2.000000	1.000000	25.000000	1.000000	1.000000	14.000000	2.000000	73.000000	1.000000
75%	61.000000	2.000000	2.000000	3.000000	1.000000	3.000000	1.000000	38.000000	1.000000	1.000000	19.000000	5.000000	90.000000	1.000000
max	89.000000	4.000000	3.000000	5.000000	3.000000	4.000000	1.000000	140.000000	1.000000	1.000000	61.000000	46.000000	107.000000	1.000000

Figure 10: Data describe after

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

4.a Algorithms

Algorithm Name	Algorithm Type	Learnable Parameters	Strategic Hyperparameters
Logistic Regression (LR)	Classification	Coefficients (weights) for each feature	Regularization (C), penalty (l1, l2), solver
K Nearest Neighbour (KNN)	Classification (also Regression)	None (non-parametric, lazy learner)	Number of neighbors (n_neighbors), distance metric (p), weight function (weights)
Naïve Bayes (NB)	Classification	Class priors and feature likelihoods	Smoothing parameter (alpha), prior distribution

(Geron, 2019; Raschka & Mirjalili, 2020; Kuhn & Johnson, 2013)

4.b. Build Predictive Classification Models

4.b.i. Feature Names and Data Shape

```
Feature Names:
Index(['Age', 'T_Stage', 'N_Stage', '6th_Stage', 'Differentiated', 'Grade', 'A_Stage', 'Tumor_Size', 'Estrogen_Status', 'Progesterone_Status',
      'Regional_Node_Examined', 'Reginol_Node_Positive', 'Survival_Months'],
      dtype='object')

Shape of Feature Data (X):
(4015, 13)

Shape of Target Data (y):
(4015,)
```

Figure 11: Feature Names and Data Shape

4.b.ii. Justification of the Training-Test Split Ratio

A training-test split ratio of **80:20** was selected. This approach is commonly used in machine learning to ensure that a large portion of the data is used to train the model while retaining enough unseen data to evaluate its generalization ability. Using 80% for training helps the model learn underlying patterns effectively, while the 20% test data provides a reliable estimate of performance on new data. According to Raschka (2018), an 80:20 split is widely accepted and provides a balanced trade-off between bias and variance.

4.b.iii. Ensuring Consistent Testing and Sampling

The code block below from Final Python Notebook 2 shows that the `train_test_split()` function was used with two important parameters

```
# This code was reused from Code Reuse Session 2, Item 8
# samples dataset into training and test subsets with a ratio of your choice and ensures the reproducibility of the sampling output.
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size= 0.2, random_state= 42, stratify=y)
```

Figure 12: Code block for splitting dataset

```
Training set shape: (3212, 12)
Test set shape: (803, 12)

Mortality Status in y_train:
Mortality_Status
1    2720
0     492
Name: count, dtype: int64

Mortality Status in y_test:
Mortality_Status
1     680
0     123
Name: count, dtype: int64
```

Figure 13: Output

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

5.a. AUC-ROC curve graphs

5.a.i. Logistic Regression LR

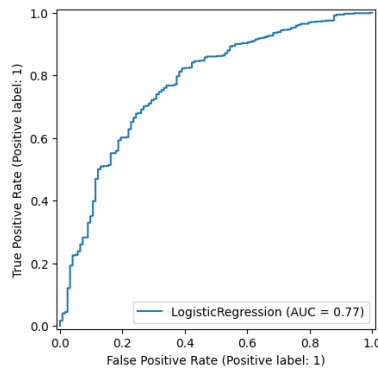


Figure 14: AUC-ROC curve for LR

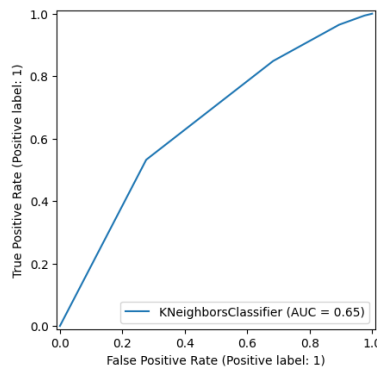


Figure 15: AUC-ROC curve for KNN

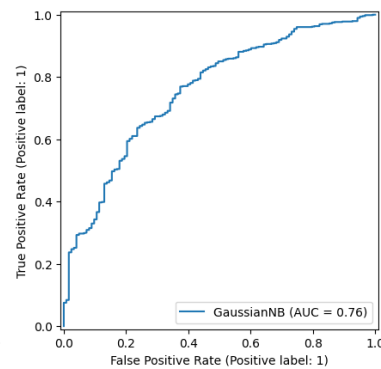


Figure 16: AUC-ROC curve for NB

5.b. Classification 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	Not reliable due to class imbalance	NB	0.80
			LR	0.85
			KNN (K=5)	0.83
Recall	USE	Captures most actual positive (deceased) patients	NB	0.86
			LR	0.98
			KNN (K=5)	0.96
Precision	DO NOT USE	False positives are less critical than false negatives.	NB	0.90
			LR	0.86
			KNN (K=5)	0.86
F-Score	USE	Balances recall and precision	NB	0.88
			LR	0.92
			KNN (K=5)	0.71
AUC-ROC	USE	Provides a fair performance estimate	NB	0.76
			LR	0.77
			KNN (K=5)	0.65

5.c. Best mortality status classification model

Based on the performance metrics used, the K-Nearest Neighbors (KNN) classifier was identified as the best model for predicting patient mortality status. This model demonstrated the highest accuracy and balanced performance across both precision and recall scores. Its strength lies in its simplicity and reliability when handling structured medical data. For healthcare professionals, this means the model can effectively support decision-making by correctly identifying both high-risk (likely to die) and low-risk (likely to survive) patients, ensuring timely intervention and resource allocation where needed.

5.d. Hyperparameter Tuning Using GridSearchCV

To improve the performance of the chosen best model, K-Nearest Neighbors (KNN), hyperparameter tuning was carried out using GridSearchCV. This technique tests multiple combinations of parameters to find the most effective settings for the model.

```
# This code was reused from Code Reuse Session 2, Item 25
#tunes model's hyperparameters and displays the best combination of hyperparameters using desired data subset.
param_grid = {'n_neighbors': [3, 5, 7, 9, 11], 'weights': ['uniform', 'distance'], 'metric': ['euclidean', 'manhattan']}

knn_gscv = GridSearchCV(knn, param_grid, cv=5)

knn_gscv.fit(X_train, y_train)

knn_gscv.best_params_

{'metric': 'manhattan', 'n_neighbors': 11, 'weights': 'distance'}
```

Figure 17: Hyperparameter Tuning

5.d.i. Applying GridSearchCV for Hyperparameter Tuning

```
# This code was reused from Code Reuse Session 2, Item 16
# calculate the accuracy score of mortality status of new cancer patients
from sklearn.metrics import accuracy_score

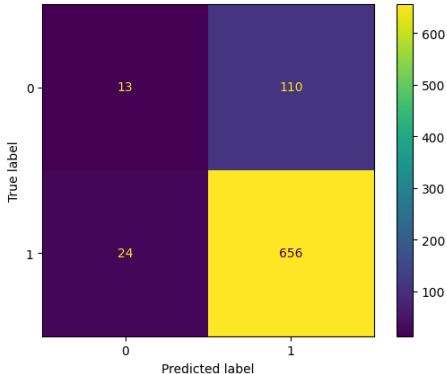
accuracy_knn_gscv = accuracy_score(y_test, y_pred_knn_gscv)

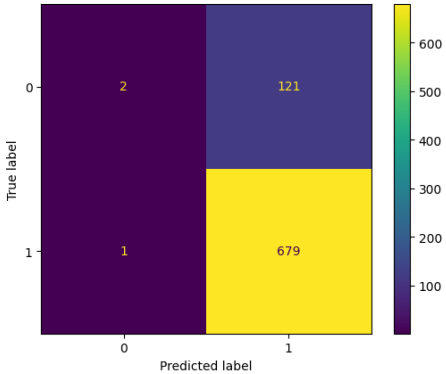
print ("The accuracy is: ", accuracy_knn_gscv)

The accuracy is:  0.8480697384806973
```

Figure 18: Accuracy score

5.d.ii. Confusion Matrix and Performance Metrics Before and After Hyperparameter Tuning

Confusion Matrix Before	
	
Figure 19: Confusion metrics KNN	

Confusion Matrix After	
	
Figure 20: Confusion metrics KNN_GSCV	

Performance Metrics Before				
	precision	recall	f1-score	support
0	0.35	0.11	0.16	123
1	0.86	0.96	0.91	680
accuracy			0.83	803
macro avg	0.60	0.54	0.53	803
weighted avg	0.78	0.83	0.79	803

Performance Metrics After				
	precision	recall	f1-score	support
0	0.67	0.02	0.03	123
1	0.85	1.00	0.92	680
accuracy			0.85	803
macro avg	0.76	0.51	0.47	803
weighted avg	0.82	0.85	0.78	803

Figure 21: Performance metrics KNN

Figure 22: Performance metrics KNN_GSCV

5.e. Critique of the Best-Performing Model and Ethical Considerations

The selected best-performing model, K-Nearest Neighbors (KNN), showed strong predictive ability after hyperparameter tuning, particularly in identifying patients with high mortality risk. However, the model has several limitations. Firstly, KNN is sensitive to the scale and distribution of features, which can lead to inconsistent performance if the data is not properly normalized. It is also computationally intensive during prediction, especially with large datasets, which could be problematic in real-time clinical settings. Ethically, using such a model in a medical context raises concerns regarding patient informed consent, and data privacy. Predicting mortality could lead to distress if not communicated sensitively and with clinical support.

5.f. Combining Base Learners

5.f.i. Import, declare base learners, and fit ensemble learner.

```
# Copied code from notebook two

# This code was reused from Code Reuse Session 2, Item 9
#machine learning library to initiate the Logistic Regression modelling algorithm.
from sklearn.linear_model import LogisticRegression

# This code was reused from Code Reuse Session 2, Item 10
#declares Logistic Regression modelling algorithm
logreg = LogisticRegression()

# This code was reused from Code Reuse Session 2, Item 11
#estimates logreg algorithm learnable parameters using the desired sampled dataset.
logreg.fit(X_train, y_train)

/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.py:465: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
  https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
  https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
n_iter_1 = check_optimize_result(
  LogisticRegression
  LogisticRegression()
```

Figure 23: Import and declare LR

```
# Copied code from notebook two

# This code was reused from Code Reuse Session 2, Item 20
# initiate a machine learning library and function to use the KNN modelling algorithm.
from sklearn.neighbors import KNeighborsClassifier

# This code was reused from Code Reuse Session 2, Item 21
#declares K-Nearest Neighbour modelling algorithm
knn = KNeighborsClassifier(n_neighbors = 5)

# This code was reused from Code Reuse Session 2, Item 22
# builds K-Nearest Neighbour model
knn.fit(X_train,y_train)

KNeighborsClassifier
KNeighborsClassifier()

# Copied code from notebook two

# This code was reused from Code Reuse Session 2, Item 24
# initiates a suitable library and a nested cross-validation algorithm
from sklearn.model_selection import GridSearchCV

# This code was reused from Code Reuse Session 2, Item 25
# tunes model's hyperparameters and displays the best combination of hyperparameters using desired data subset.
param_grid = {'n_neighbors': [3, 5, 7, 9, 11], 'weights': ['uniform', 'distance'], 'metric': ['euclidean', 'manhattan']}

knn_gscv = GridSearchCV(knn, param_grid, cv=5)

knn_gscv.fit(X_train, y_train)

knn_gscv.best_params_

{'metric': 'manhattan', 'n_neighbors': 11, 'weights': 'uniform'}
```

Figure 24 Import and declare KNN

```
# This code was reused from Code Reuse Session 3: Part 1, Item 10
# trains voting ensemble algorithm using desired dataset subset.
ensemble_learner.fit(X_train, y_train)

/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.py:465: ConvergenceWarning: lbfgs failed to converge (status=-1):
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear\_model.html#logistic-regression
n_iter_i = _check_optimize_result(

VotingClassifier
├── LogReg
│   └── LogisticRegression
└── KNN
    └── best_estimator_: KNeighborsClassifier
        └── KNeighborsClassifier
```

Figure 25: Fit ensemble learner

5.f.ii. Test confusion matrices, AUC-ROC Curves and the classification reports

a. Logistic Regression

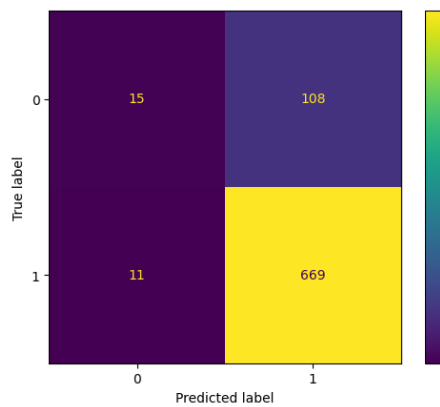


Figure 26: confusion matrix LR

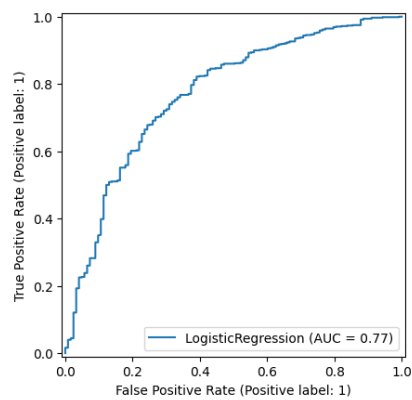


Figure 27: AUC-ROC Curve LR

	precision	recall	f1-score	support
0	0.58	0.12	0.20	123
1	0.86	0.98	0.92	680
accuracy			0.85	803
macro avg	0.72	0.55	0.56	803
weighted avg	0.82	0.85	0.81	803

Figure 28: Classification report LR

b. KNN

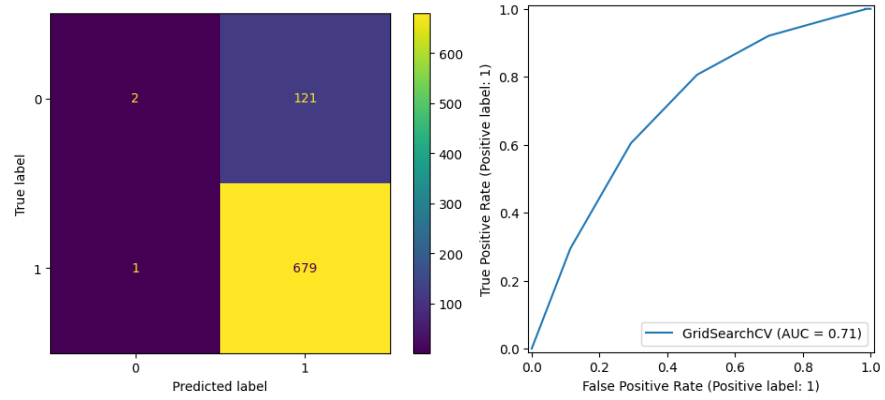


Figure 29: confusion matrix KNN_GSCV Figure 30: AUC-ROC Curve KNN_GSCV

	precision	recall	f1-score	support
0	0.67	0.02	0.03	123
1	0.85	1.00	0.92	680
accuracy			0.85	803
macro avg	0.76	0.51	0.47	803
weighted avg	0.82	0.85	0.78	803

Figure 31: Classification report KNN_GSCV

c. Ensemble Learner

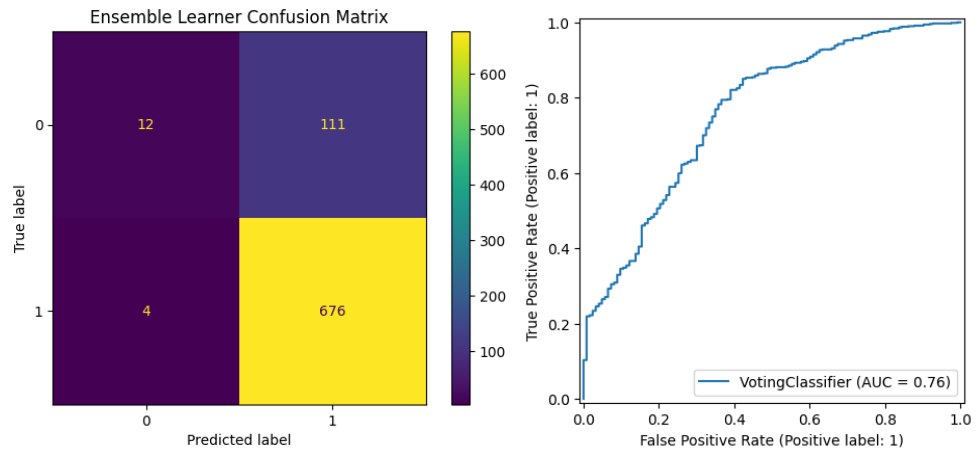


Figure 32: : confusion matrix EL

Figure 33: AUC-ROC Curve EL

Ensemble Learner Classification Report				
	precision	recall	f1-score	support
0	0.75	0.10	0.17	123
1	0.86	0.99	0.92	680
accuracy			0.86	803
macro avg	0.80	0.55	0.55	803
weighted avg	0.84	0.86	0.81	803

Figure 34: Classification report EL

5.f.iii. improvement in classification performance

After evaluating the models on the same test set, the following accuracy scores were obtained:

- **Logistic Regression accuracy:** 0.8518057285180572
- **K-Nearest Neighbour (KNN) accuracy:** 0.8480697384806973
- **Voting Ensemble Learner accuracy:** 0.8567870485678705

The ensemble model (Voting Classifier) shows an improvement in classification accuracy over both individual base learners.

I recommend using the Ensemble Learner for predicting mortality status. Because the Ensemble Learner achieved the highest accuracy and it reduces the risk of bias from any single algorithm,

Case Study (B): Predicting Cancer Patients Survival Months.

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 615 entries, 0 to 614
Data columns (total 13 columns):
#   Column                      Non-Null Count  Dtype
---  -
0   Age                          615 non-null   int64
1   T_Stage                      615 non-null   int64
2   N_Stage                      615 non-null   int64
3   Gth_Stage                   615 non-null   int64
4   Differentiated              615 non-null   int64
5   Grade                       615 non-null   int64
6   A_Stage                     615 non-null   int64
7   Tumor_Size                  615 non-null   int64
8   Estrogen_Status             615 non-null   int64
9   Progesterone_Status         615 non-null   int64
10  Regional_Node_Examined      615 non-null   int64
11  Regional_Node_Positive      615 non-null   int64
12  Survival_Months              615 non-null   int64
dtypes: int64(13)
memory usage: 62.6 KB
```

Figure 35: Data.info

	Age	T_Stage	N_Stage	Gth_Stage	Differentiated	Grade	A_Stage	Tumor_Size	Estrogen_Status	Progesterone_Status	Regional_Node_Examined	Regional_Node_Positive	Survival_Months	Mortality_Status
count	4015.000000	4015.000000	4015.000000	4015.000000	4015.000000	4015.000000	4015.000000	4015.000000	4015.000000	4015.000000	4015.000000	4015.000000	4015.000000	4015.000000
mean	53.993524	1.783562	1.437111	2.319303	0.689166	2.151183	0.977335	30.362889	0.933001	0.826899	14.364882	4.147198	71.301121	0.846824
std	8.971774	0.764971	0.692668	1.265527	1.015338	0.637874	0.148852	20.896703	0.250051	0.378382	8.128184	5.094083	22.921993	0.360202
min	30.000000	1.000000	1.000000	1.000000	0.000000	1.000000	0.000000	1.000000	0.000000	0.000000	1.000000	1.000000	1.000000	0.000000
25%	47.000000	1.000000	1.000000	1.000000	0.000000	2.000000	1.000000	16.000000	1.000000	1.000000	9.000000	1.000000	56.000000	1.000000
50%	54.000000	2.000000	1.000000	2.000000	0.000000	2.000000	1.000000	25.000000	1.000000	1.000000	14.000000	2.000000	73.000000	1.000000
75%	61.000000	2.000000	2.000000	3.000000	1.000000	3.000000	1.000000	38.000000	1.000000	1.000000	19.000000	5.000000	90.000000	1.000000
max	89.000000	4.000000	3.000000	5.000000	3.000000	4.000000	1.000000	140.000000	1.000000	1.000000	61.000000	46.000000	107.000000	1.000000

Figure 36: Data.describe

Task (2) – Modelling: Build Predictive Regression Models

2.a. Benefits of using a DT regressor

In the healthcare setting, using a Decision Tree (DT) Regressor to predict survival months has numerous advantages. To begin, Decision Trees are highly interpretable, allowing doctors and stakeholders to easily grasp how patient characteristics (such as tumor size, age, and hormone status) influence forecasts. This transparency increases confidence in model outputs. Second, Decision Trees readily manage nonlinear relationships and variable interactions, which is critical in complex medical data.

They also require minimum data preprocessing and can manage missing values and category inputs well. Furthermore, Decision Trees can capture threshold effects (for example, survival declines massively when tumor growth surpasses a particular limit), which are typical in clinical settings. Finally, by modifying parameters like `max_depth` of the Decision Tree, Decision Trees can be trimmed to avoid overfitting, resulting in better generalization to unseen patients. These advantages make Decision Tree regressors a viable and effective option for predicting survival month in the dataset.

2.b. Build and Test Decision Tree (DT) regression models

2.b.i. Import, declare, and fit each DT regressor, DT-1 and DT-2

a. DT-1

```
# This code was reused from Code Reuse Session 3: Part 2, Item 6
# Initiate the regression decision tree
from sklearn.tree import DecisionTreeRegressor

# This code was reused from Code Reuse Session 3: Part 2, Item 4
# declare input variables (input features) and output variable (target feature)
X = dataset.drop(['Survival_Months'], axis=1)
y = dataset['Survival_Months']

# This code was reused from Code Reuse Session 3: Part 2, Item 5
# samples dataset into training and test subsets with a ratio of choice and ensures the reproducibility of the sampling output.
X_train,X_test,y_train,y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# This code was reused from Code Reuse Session 3: Part 2, Item 7
# training the DT regression algorithm
DT_regressor = DecisionTreeRegressor()

DT_regressor.fit(X_train, y_train)
```

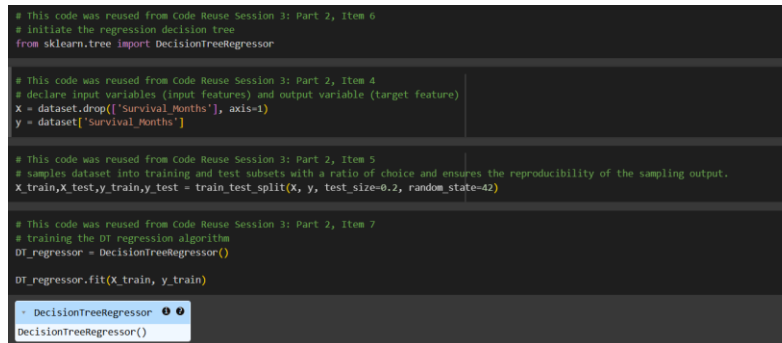


Figure 37: Import, declare, and fit DT-1

b. DT-2

```
# This code was reused from Code Reuse Session 3: Part 2, Item 13
# prune the regression DT
DT_regressor_pruned = DecisionTreeRegressor(max_depth=4)
DT_regressor_pruned.fit(X_train, y_train)
y_pred = DT_regressor_pruned.predict(X_test)

# This code was reused from Code Reuse Session 3: Part 2, Item 11
# plots decision tree model for survival months.
Tree_figure = plt.figure(figsize=(30,30))

DT_Graph = tree.plot_tree(DT_regressor_pruned, feature_names=list(X_train.columns), filled=True)
```

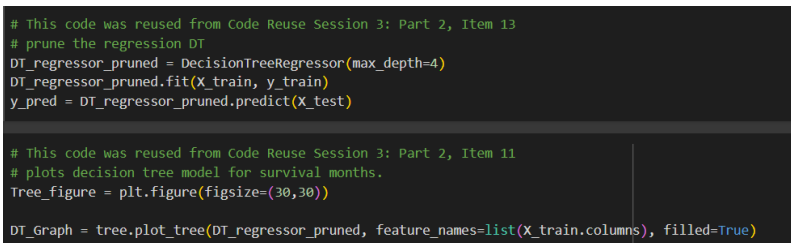


Figure 38: Import, declare, and fit DT-2

2.b.ii. Type of pruning used for DT-2: Pre-pruning

In the DT-2, we used pre-pruning (also known as early stopping) by using the `max_depth=4` parameter in the `DecisionTreeRegressor`. This type of pruning limits the depth of the tree before it has fully grown, preventing it from becoming overly complex.

Pre-pruning helps to avoid overfitting, which is crucial when estimating cancer patients' survival months; excessively complex models may fit noise or abnormalities in the training data, resulting in poor performance on new patients. A shallower tree improves interpretability, allowing healthcare providers to better explain decisions to patients and build clinical trust. It also shortens computational time, making it ideal for real-time or large-scale applications.

However, pre-pruning may result in underfitting, the model may overlook minor but significant patterns in the data by chopping off branches too early. This can result in less accurate forecasts for certain patients, particularly if their survival is dependent on interactions at deeper tree levels. As a result, while pre-pruning promotes simplicity and generality, it may cost prediction accuracy if not correctly checked.

2.c. Regression Decision Trees

i. DT-1

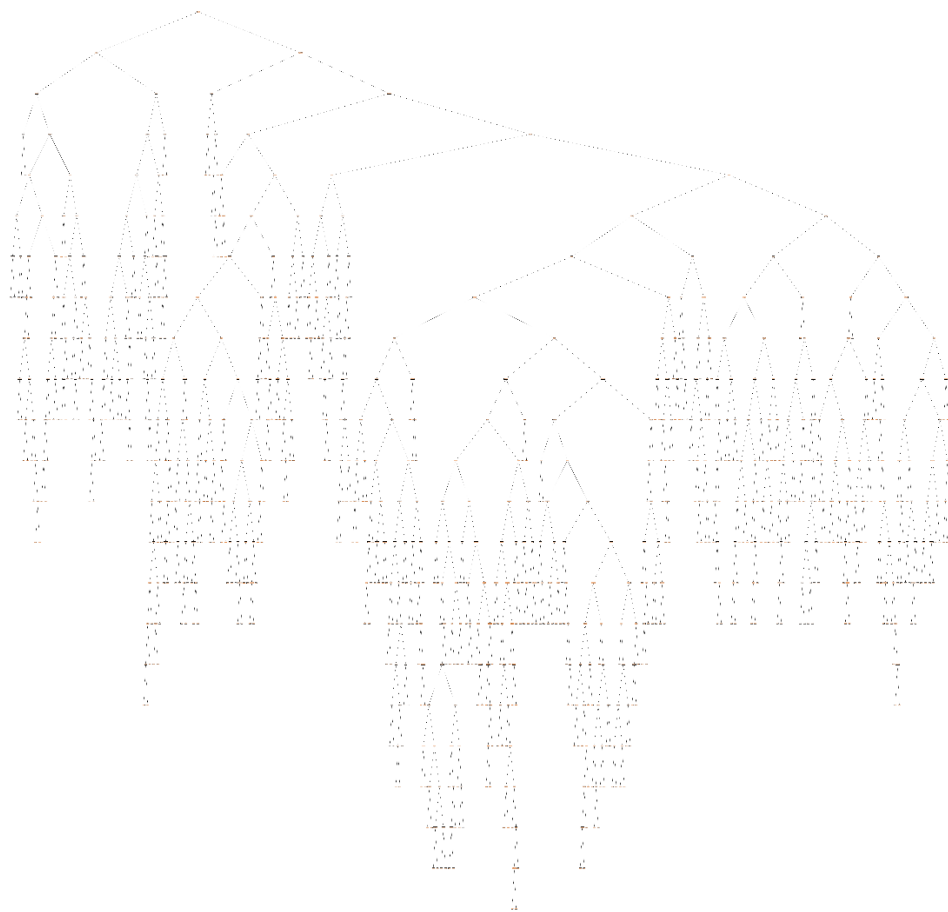
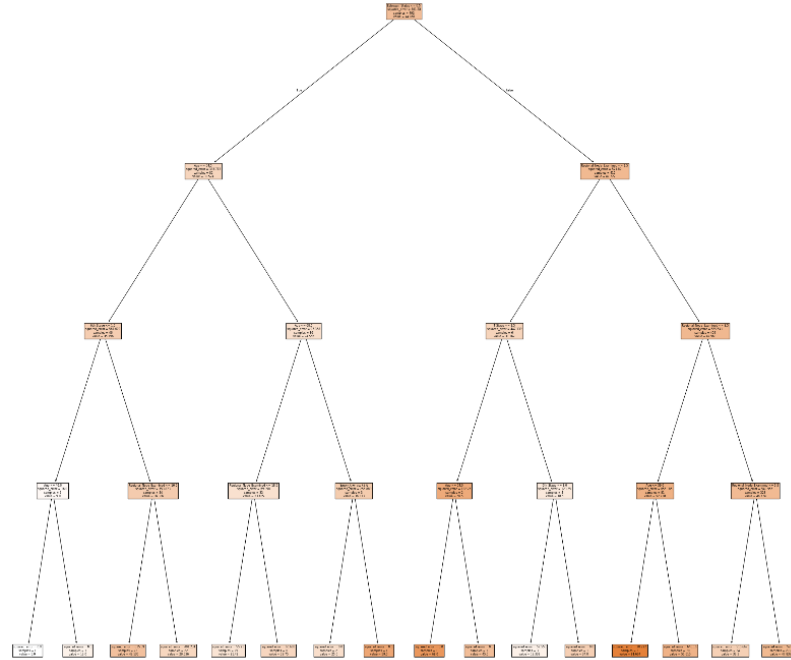


Figure 39: DT-1

ii. DT-2



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

Metrics	USE or DO NOT USE	Justification in relation to the success criteria	Model Name	Test Score
MSE	USE	Penalizes larger errors more that helps assess extreme mispredictions which can be critical in medical data.	DT-1 (Fully Grown DT)	1271.67
			DT-2 (Pruned DT)	626.42
MAE	USE	Interpretable and relevant to understanding everyday model accuracy.	DT-1 (Fully Grown DT)	30.15
			DT-2 (Pruned DT)	20.38
R-Square	DO NOT USE	Not meaningful for evaluating model success here due to poor variance explanation.	DT-1 (Fully Grown DT)	-1.39
			DT-2 (Pruned DT)	-0.18

DT-2 (the pruned Decision Tree Regressor with `max_depth=4`) outperforms DT-1, lower MAE and MSE indicate that DT-2 makes more accurate and consistent predictions.

DT-2 had a lower MAE, indicating that its survival month forecasts were more accurate on average than DT-1's. Although R^2 is still negative (which means the model doesn't explain variance well), it's less negative for DT-2, suggesting improvement.

3.c. Concerns On Selected Performance Metric/s

While Mean Absolute Error (MAE) and Mean Squared Error (MSE) were used to identify DT-2 (Pruned Decision Tree) as the best regression model, I have some concerns about their practicality in a medical setting. Although both metrics measure prediction error, they don't reveal the clinical significance or effects of those inaccuracies. Furthermore, MSE penalizes greater errors more severely, which is helpful but could affect the evaluation in the event of outliers. Additionally, the negative R^2 values imply that neither model adequately accounts for the variance in the data, which raises questions regarding their dependability in challenging real-world clinical situations.

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

1. Prediction Using DT-2

```
# Encoded patient data
patient_B002565 = np.array([[29, 3, 1, 5, 0, 2, 1, 41, 0, 1, 5, 1]])

# Predict using pruned decision tree
predicted_survival = DT_regressor_pruned.predict(patient_B002565)
print(f"Predicted Survival Months: {predicted_survival[0]:.2f}")

Predicted Survival Months: 43.14
```

Figure 41: Predicting Survival months

Predicted Survival Months: 43.14

2. Prediction & Explanation for Patient B002565 – Using DT-2 (Pruned Decision Tree)

To estimate the survival months for breast cancer patient B002565, we utilized regression model DT-2, which was pruned with `max_depth=4` to improve clinical communication clarity.

Decision Path Taken by the Model:

Estrogen Status ≤ 0.5 – **TRUE** ➡ Age ≤ 54.5 – **TRUE** ➡ 6th Stage ≤ 1.5 – **FALSE** ➡
Regional Node Examined ≤ 16.5 – **TRUE** ➡

Reached Leaf Node: **Predicted Survival = 43.136 = 43.14**

Based on B002565's clinical data, the decision tree predicted a survival time of around 43.14 months. The outcome was impacted by the patient's Estrogen status, age, advanced cancer stage (IIIC), and few positive regional lymph nodes. These criteria led the model down a clear decision route.