UNIVERSITY OF WESTMINSTER

Informatics Institute of Technology

Department of Computing

5DATA001C.2 Machine Learning and Data Mining

Name: Ravindi Madana

UoW Number: w2082268

IIT Number: 20232777

Group Number: 23

Contents

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	Don't need for predictions
Month of Birth	DROP	No medical relevant
Age	RETAIN	Important risk fact, according to age possibility check
Sex	DROP	All patient here seems like female
Occupation	DROP	So many types of occupation can't encode
T Stage	RETAIN	Tumor stage is a critical predictor for survival rate
N Stage	RETAIN	Nodal status is a key indicator of breast cancer outcomes
6th Stage	RETAIN	Combined staging system provides comprehensive disease progression assessment
Differentiated	RETAIN	Tumor differentiation is a prognostic factor in breast cancer
Grade	RETAIN	Histological grade is strongly associated with patient survival
A Stage	RETAIN	Reports whether cancer has spread to distant locations, a major determinant of death
Tumor Size	RETAIN	Tumor size is directly related to risk of death
Estrogen Status	RETAIN	Estrogen receptor status affects response to therapy and prognosis
Progesterone Status	RETAIN	Progesterone receptor status affects treatment decisions and survival

Regional Node Examined	RETAIN	Number of nodes assessed is related to extent of disease assessment
Regional Node Positive	RETAIN	Number of positive nodes is one of the strongest predictors of recurrence and survival
Survival Months	DROP	This is post-outcome data which would lead to leakage of data in mortality prediction
Mortality Status	RETAIN	This is our target variable for the classification problem.

Task (2) – Exploring and Understanding Your Dataset

Variable Data Types

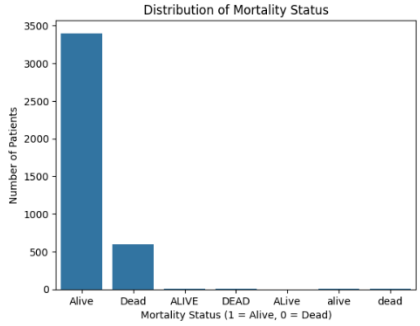
```
data_cleaned.info()

<class 'pandas.core.frame.DataFrame'>
Index: 4020 entries, 0 to 4023
Data columns (total 14 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Age                     4020 non-null   Int64
 1   T_Stage                 4020 non-null   int64
 2   N_Stage                 4020 non-null   int64
 3   6th_Stage               4020 non-null   int64
 4   Differentiated          4020 non-null   int64
 5   Grade                   4020 non-null   Int64
 6   A_Stage                 4020 non-null   int64
 7   Tumor_Size              4020 non-null   Int64
 8   Estrogen_Status         4020 non-null   int64
 9   Progesterone_Status     4020 non-null   int64
 10  Regional_Node_Examined  4020 non-null   Int64
 11  Regional_Node_Positive  4020 non-null   Int64
 12  Survival_Months         4020 non-null   Int64
 13  Mortality_Status        4020 non-null   int64
dtypes: Int64(6), int64(8)
memory usage: 494.6 KB
```

Descriptive Statistics Table

```
data_cleaned.describe()
```

	Age	T_Stage	N_Stage	6th_Stage	Differentiated	Grade	A_Stage	Tumor_Size	Estrogen_Status	Progesterone_Status	Regional_Node_Examined	Regional_Node_Positive	Survival_Months	Mortality_Status
count	4020.0	4020.000000	4020.000000	4020.000000	4020.000000	4020.0	4020.000000	4020.0	4020.000000	4020.000000	4020.0	4020.0	4020.0	4020.000000
mean	53.992537	0.784826	0.437562	2.320647	1.150995	2.150995	0.022637	30.454229	0.933085	0.826617	14.367413	4.149751	71.476866	0.847015
std	8.971573	0.765464	0.692858	1.265627	0.638280	0.63828	0.148761	21.094478	0.249907	0.378625	8.127259	5.093743	25.361506	0.360018
min	30.0	0.000000	0.000000	1.000000	0.000000	1.0	0.000000	1.0	0.000000	0.000000	1.0	1.0	1.0	0.000000
25%	47.0	0.000000	0.000000	1.000000	1.000000	2.0	0.000000	16.0	1.000000	1.000000	9.0	1.0	56.0	1.000000
50%	54.0	1.000000	0.000000	2.000000	1.000000	2.0	0.000000	25.0	1.000000	1.000000	14.0	2.0	73.0	1.000000
75%	61.0	1.000000	1.000000	3.000000	2.000000	3.0	0.000000	38.0	1.000000	1.000000	19.0	5.0	90.0	1.000000
max	89.0	3.000000	2.000000	5.000000	3.000000	4.0	1.000000	140.0	1.000000	1.000000	61.0	46.0	760.0	1.000000

Mortality_Status Distribution Plot



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

3 - (a)

Variable Name	Issue found	Proposed fix	Justification for used fix method
Reginol_Node_Positive	Misspelled column name	Rename to Regional_Node_Positive	Readability and correctness are improved.
Mortality_Status	Inconsistent capitalization and spelling (ALIVE, alive, Dead, DEAD)	Convert to lowercase, strip whitespace, map values to Alive and Dead	Standardizes categories for binary classification
Age, Tumor_Size	Contains invalid entries (Minus values) and out of range values	Convert to numeric with errors='coerce', remove out-of-ange	Invalid entries and outliers may bias or crash models.
Missing Columns	Contain missing values	Impute using mean	Mean imputation prevents row loss while being effective continuous data default.
T_Stage, N_Stage, Grade, Differentiated	Ordinal categorical variables not in numerical form	Encode using LabelEncoder or mapping	Models can't handle strings. Encoding ordinal categories preserves the natural order,
Estrogen_Status, Progesterone_Status, A_Stage, 6th_Stage	Categorical string labels	Map values to numeric manually	Categorical strings must be numerically encoded for ML models.
Sex, Occupation	Irrelevant or redundant variables	Drop columns	These are not likely to be informative for mortality prediction
Float columns with no decimals	Some float columns don't actually contain fractional values	Convert to Int64	Preserves compatibility with models expecting integers.

3 - (b)

Before Fix (Regional_Node_Positive):

```
print(data.columns)

upation', 'T_Stage', 'N_Stage', '6th_Stage', 'Differentiated', 'Grade', 'A_S
tus', 'Regional_Node_Examined', 'Reginol_Node_Positive', 'Survival_Months',
```

After Fix (Regional_Node_Positive):

```
print(data_cleaned.columns)

Index(['Age', 'T_Stage', 'N_Stage', '6th_Stage', 'Differentiated', 'Grade', 'A_Sta
       'Regional_Node_Examined', 'Regional_Node_Positive', 'Survival_Months', 'Mor
      dtype='object')
```

Before Fix (Mortality_Status):

```
# BEFORE FIX: Show inconsistent Mortality_Status values
print(data['Mortality_Status'].unique())

['Alive' 'Dead' 'ALIVE' 'DEAD' 'ALive' 'alive' 'dead']
```

After Fix (Mortality_Status):

```
[ ]  data_cleaned['Mortality_Status'].unique()

     array([1, 0])
```

Before Fix (Age and Tumor_Size):

```
data['Age'].unique()

array([ 68.,  50.,  58.,  47.,  51.,  40.,  69.,  46.,  65.,  48.,  62.,
        61.,  56.,  43.,  60.,  57.,  55.,  63.,  66.,  53.,  59.,  54.,
        49.,  64.,  42.,  nan,  37.,  67.,  31.,  52.,  33.,  45.,  38.,
        39.,  36., 180.,  41.,  44., -50.,  32.,  34., 502.,  35.,  30.,
        89.])
```

```
[101] data['Tumor_Size'].unique()

array([  4.,  35.,  63.,  18.,  41.,  20.,   8.,  30., 103.,  32.,  13.,
        59.,  15.,  19.,  46.,  24.,  25.,  29.,  40.,  70.,  22.,  50.,
        17.,  21.,  10.,  27.,  23.,   5.,  51.,   9.,  55., 120.,  77.,
         2.,  11.,  12.,  26.,  75., 130.,  34.,  80.,   3.,  60.,  14.,
        16.,  45.,  36.,  76.,  38.,  49.,   7.,  72., 100.,  43.,  62.,
        37.,  68., -75.,  52.,  85.,  57.,  39.,  28.,  48., 110.,  65.,
         6., 105., 140.,  42.,  31.,  90., 108.,  98.,  47.,  54.,  61.,
        74.,  33.,   1.,  87.,  nan,  81.,  58., 117.,  44., 123., 133.,
        95., 107.,  92.,  69.,  56.,  82.,  66.,  78.,  97.,  88.,  53.,
        83., 101.,  84., 115.,  73., 125., 104.,  94.,  86.,  64.,  96.,
        79.,  67.])
```

After Fix (Age and Tumor_Size):

```
data_cleaned['Age'].unique()

<IntegerArray>
[68, 50, 58, 47, 51, 40, 69, 46, 65, 48, 62, 61, 56, 43, 60, 57, 55, 63, 66, 53,
59, 54, 49, 64, 42, 37, 67, 31, 52, 33, 45, 38, 39, 36, 41, 44, 32,
 34, 35, 30, 89]
Length: 41, dtype: Int64
```

```
data_cleaned['Tumor_Size'].unique()

<IntegerArray>
[  4,  35,  63,  18,  41,  20,   8,  30, 103,  32,
 ...
 115,  73, 125, 104,  94,  86,  64,  96,  79,  67]
Length: 110, dtype: Int64
```

Before Fix (Missing Columns):

	0
Patient_ID	0
Month_of_Birth	0
Age	9
Sex	4
Occupation	3981
T_Stage	0
N_Stage	0
6th_Stage	0
Differentiated	0
Grade	0
A_Stage	0
Tumor_Size	3
Estrogen_Status	0
Progesterone_Status	0
Regional_Node_Examined	1
Reginol_Node_Positive	0
Survival_Months	0
Mortality_Status	0

dtype: int64

After Fix (Missing Columns):

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

dtype: int64

Before Fix (Data Encoding and Float Converting)

```
Data type of all variables:
<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  4023 non-null   float64
 15  Reginol_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
```

After Fix

```
Data type of all variables:
<class 'pandas.core.frame.DataFrame'>
Index: 4020 entries, 0 to 4023
Data columns (total 14 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Age                     4020 non-null   Int64
 1   T_Stage                 4020 non-null   int64
 2   N_Stage                 4020 non-null   int64
 3   6th_Stage               4020 non-null   int64
 4   Differentiated          4020 non-null   int64
 5   Grade                   4020 non-null   Int64
 6   A_Stage                 4020 non-null   int64
 7   Tumor_Size              4020 non-null   Int64
 8   Estrogen_Status         4020 non-null   int64
 9   Progesterone_Status     4020 non-null   int64
 10  Regional_Node_Examined  4020 non-null   Int64
 11  Regional_Node_Positive  4020 non-null   int64
 12  Survival_Months         4020 non-null   Int64
 13  Mortality_Status        4020 non-null   int64
dtypes: Int64(6), int64(8)
memory usage: 494.6 KB
```

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

4 - (a)

Algorithm Name	Algorithm Type	Learnable Parameters	Some Strategic Hyperparameters
NB	Parametric	Class prior probabilities,	None (typically not tunable in GaussianNB)

		likelihoods (mean, variance for Gaussian NB)	
LR	Parametric	Weights/coefficients (β values)	Regularization (C), penalty (L1/L2), solver
KNN (N=?)	Non-parametric	None (lazy learner)	Number of neighbors (k), distance metric, weights

4 - (b)

4 - (b) - i

```
print(X_train.columns)
print(X_train.shape, X_test.shape)
```

```
Index(['Age', 'T_Stage', 'N_Stage', '6th_Stage', 'Differentiated', 'Grade',
       'A_Stage', 'Tumor_Size', 'Estrogen_Status', 'Progesterone_Status',
       'Regional_Node_Examined', 'Regional_Node_Positive', 'Mortality_Status'],
      dtype='object')
(430, 13) (185, 13)
```

4 - (b) -ii

Here employed an 80/20 train-test split to ensure most data is left for model learning and a significant amount is set aside for safe testing. Such balance allows us to build models with satisfactory generalization performance. The dataset is not extremely small nor highly imbalanced, thus the application of an 80/20 split in sound model evaluation.

4 - (b) - iii

```
[8]  # This code from code reuse session 3 pt 7
     # Split the data into training and testing sets
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42, stratify=y)
```

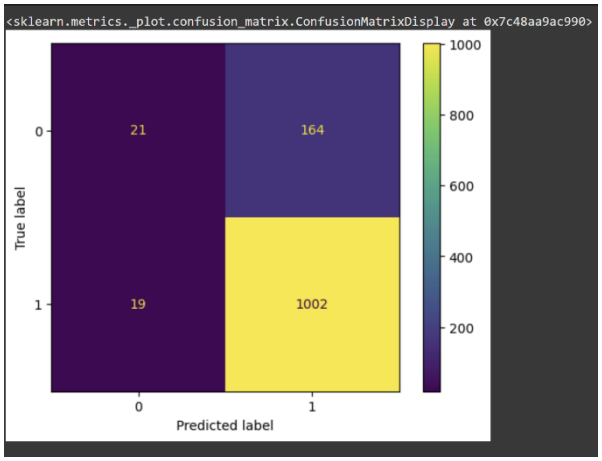
Here used the stratify = y parameter to maintain class balance between "Alive" and "Dead" cancer patients both in training and test sets. This is necessary for unbiased comparison of classification performance. We have also used random_state=42 to make results reproducible and all models get tested on same data subset.

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

5 - (a)

I. Logistic Regression (LR)

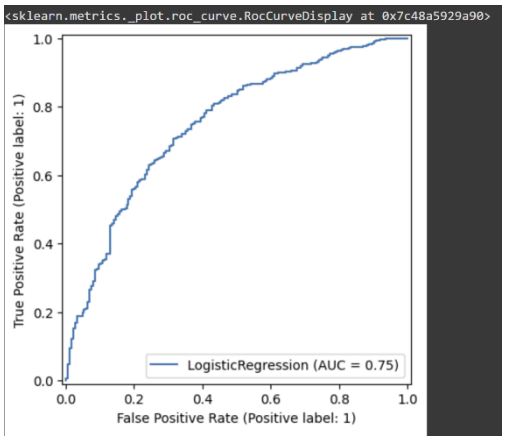
a) Confusion Matrix

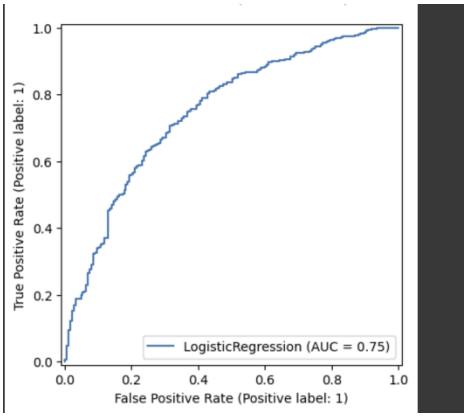


b) Classification Report

	precision	recall	f1-score	support
0	0.53	0.11	0.19	185
1	0.86	0.98	0.92	1021
accuracy			0.85	1206
macro avg	0.69	0.55	0.55	1206
weighted avg	0.81	0.85	0.80	1206

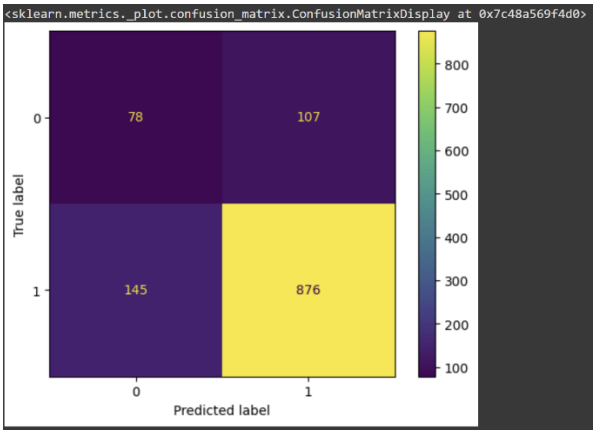
c)AUC-ROC Curve





II. Naive Bayes (NB)

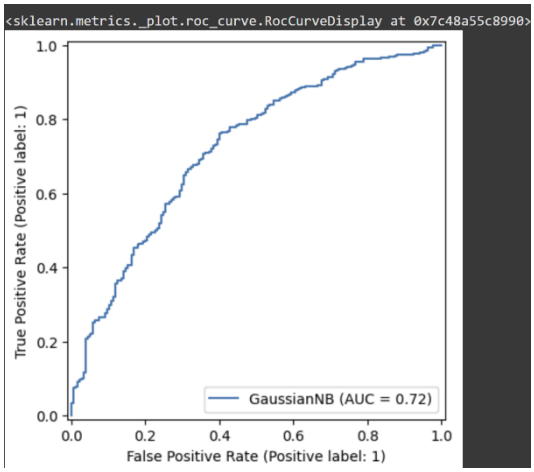
a) Confusion Matrix

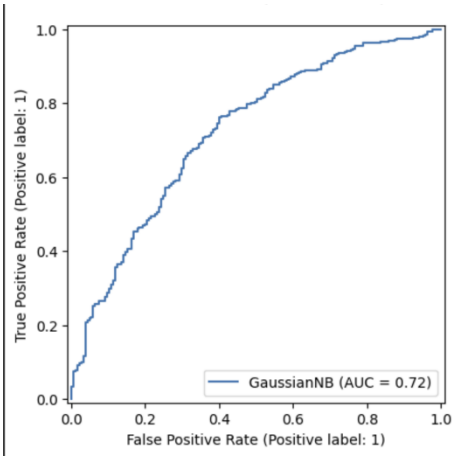


b) Classification Report

	precision	recall	f1-score	support
0	0.35	0.42	0.38	185
1	0.89	0.86	0.87	1021
accuracy			0.79	1206
macro avg	0.62	0.64	0.63	1206
weighted avg	0.81	0.79	0.80	1206

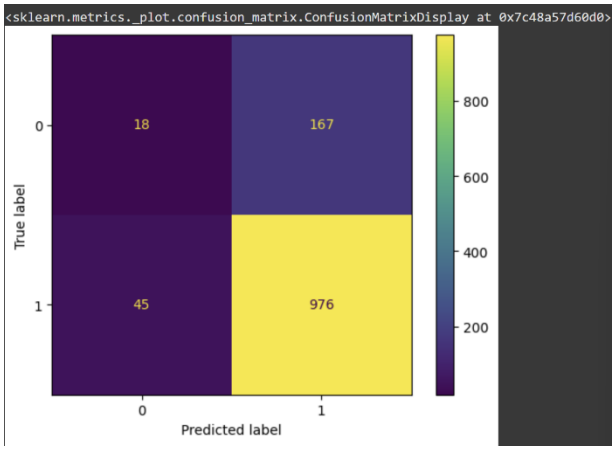
c)AUC-ROC Curve





III. K-Nearest Neighbors (KNN)

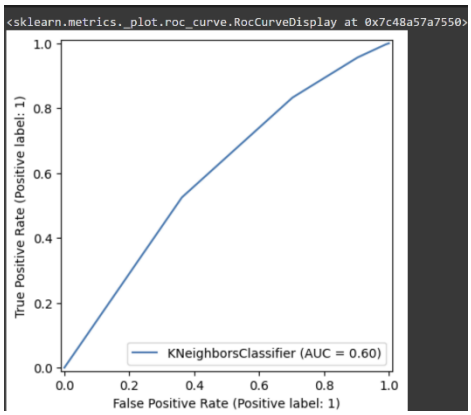
a) Confusion Matrix

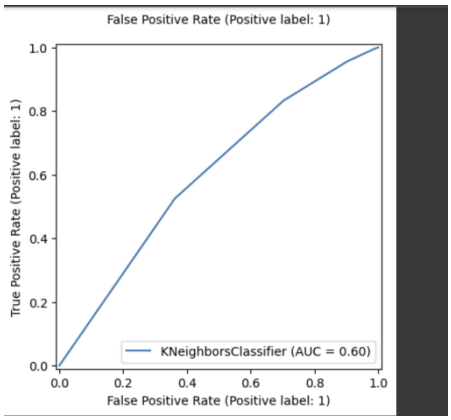


b) Classification Report

	precision	recall	f1-score	support
0	0.29	0.10	0.15	185
1	0.85	0.96	0.90	1021
accuracy			0.82	1206
macro avg	0.57	0.53	0.52	1206
weighted avg	0.77	0.82	0.79	1206

c) AUC-ROC Curve





5- (b)

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	It doesn't consider class imbalance or specific errors	NB	0.7910447761194029
			LR	0.8482587064676617
			KNN (N=?)	0.824212271973466
Recall	USE	Important to minimize false negatives	NB	0.8579823702252694
			LR	0.9813907933398629
			KNN (N=?)	0.9559255631733594
Precision	DO NOT USE	Less important than Recall in this health application	NB	0.8911495422177009
			LR	0.8593481989708405
			KNN (N=?)	0.8538932633420823
F-Score	USE	Balances recall and precision,	NB	0.874251497005988
			LR	0.9163237311385459

		useful for performance	KNN (N=?)	0.9020332717190388
AUC-ROC	USE	Measures overall class separation across thresholds	NB	0.7226142891177172
			LR	0.749180718426556
			KNN (N=?)	0.6002488286523546

5 - (c)

Based on the successful criteria provided, Logistic Regression is the best-performing model. It achieved the highest recall (0.9814), ensuring minimal false negatives in predicting “Dead” patients. It also has a strong F1 Score (0.9163) and the best AUC-ROC value (0.7492), indicating strong class discrimination. These metrics are mostly aligned with the healthcare professionals’ requirement for a reliable classifier to distinguish between mortality statuses.

5 - (d)

5 - (d) – i Evidence of specifying parameters and applying GridSearchCV

For Logistic Regression model

```
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.model_selection import StratifiedKFold


# Scale the features and do grid search
pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('lr', LogisticRegression(random_state=42, max_iter=500))
])

param_grid_lr = {
    'lr__C': [0.01, 0.1, 1, 10, 100],
    'lr__solver': ['liblinear', 'lbfgs']
}

grid_lr = GridSearchCV(pipeline, param_grid_lr, cv=3, scoring='f1_macro')
grid_lr.fit(X_train, y_train)

print("Best parameters:", grid_lr.best_params_)
best_lr_model = grid_lr.best_estimator_
```

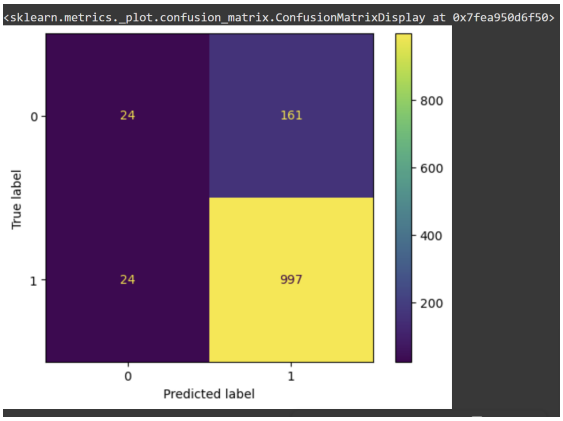
```
Best parameters: {'lr__C': 0.01, 'lr__solver': 'liblinear'}
```

```
# AFTER TUNING CONFUSION MATRIX & METRICS
print("\nAfter Hyperparameter Tuning Metrics:")
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_tuned))
print("Classification Report:\n", classification_report(y_test, y_pred_tuned))
print("AUC-ROC:", roc_auc_score(y_test, y_proba_tuned))
```

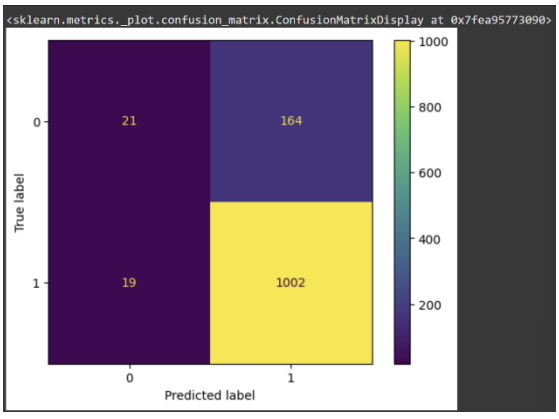
```
After Hyperparameter Tuning Metrics:
Confusion Matrix:
 [[  20  165]
 [  16 1005]]
Classification Report:
              precision    recall  f1-score   support

           0       0.56      0.11      0.18       185
           1       0.86      0.98      0.92      1021

    accuracy                           0.85      1206
   macro avg       0.71      0.55      0.55      1206
weighted avg       0.81      0.85      0.80      1206

AUC-ROC: 0.7555020250416921
```

5 - (d) – ii Confusion matrix and metrics before and after tuning

a) After



b) Before



	precision	recall	f1-score	support
0	0.53	0.11	0.19	185
1	0.86	0.98	0.92	1021
accuracy			0.85	1206
macro avg	0.69	0.55	0.55	1206
weighted avg	0.81	0.85	0.80	1206

	precision	recall	f1-score	support
0	0.50	0.13	0.21	185
1	0.86	0.98	0.92	1021
accuracy			0.85	1206
macro avg	0.68	0.55	0.56	1206
weighted avg	0.81	0.85	0.81	1206

5 - (e) While the Logistic Regression model performed well in mortality status prediction, it has the limitation of assuming linearity between log-odds and features, which may limit its capacity to recognize complex patterns. Mortality class imbalance may also bias the model. Ethically, risks are the potential for false positives/negatives to influence medical decisions. Therefore, predictions must support, not replace, expert opinion, and patient data confidentiality must be maintained at all costs.

5 - (f) – i

Python code block for import, declare base learners, and fit ensemble learner

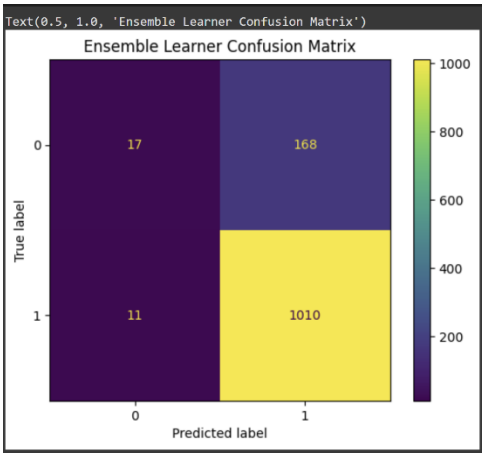
```
# This code from code reuse session 3 pt 8
# Import VotingClassifier for ensemble
from sklearn.ensemble import VotingClassifier

# This code from code reuse session 3 pt 9
# Import base learners
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier

# Define base learners
logreg = LogisticRegression()
knn = KNeighborsClassifier()
base_learners = [('LogisticRegression', logreg), ('KNN', knn)]

# Define soft voting ensemble
ensemble_learner = VotingClassifier(estimators=base_learners, voting='soft')
```

```
# This code from code reuse session 3 pt 10
# Fit the ensemble model
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
LogisticRegression
KNN
LogisticRegression
KNeighborsClassifier
```

5 - (f) – ii

a) Ensemble Learner

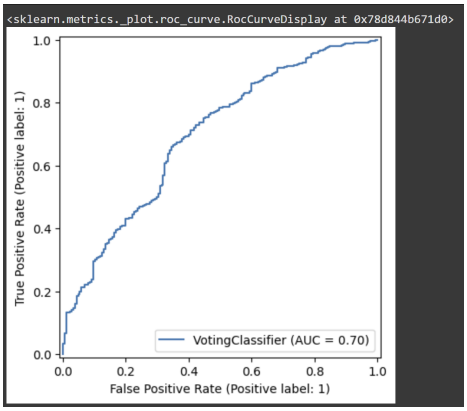
a) Confusion Matrix



b) Classification Report

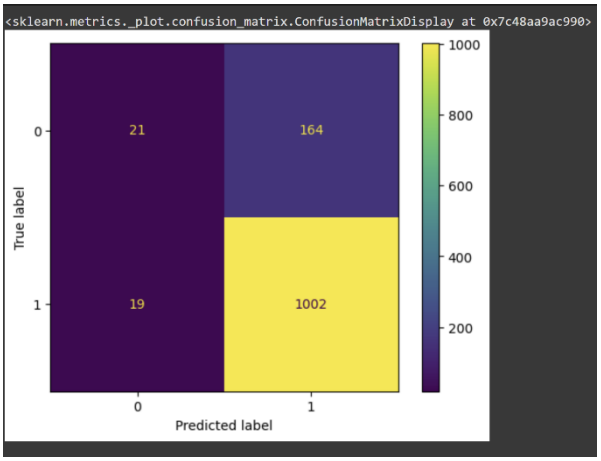
```
Ensemble Learner Classification Report
              precision    recall  f1-score   support

           0       0.61      0.09      0.16       185
           1       0.86      0.99      0.92      1021

    accuracy                           0.85      1206
   macro avg       0.73      0.54      0.54      1206
weighted avg       0.82      0.85      0.80      1206
```

c) AUC-ROC Curve



Logistic Regression (LR)

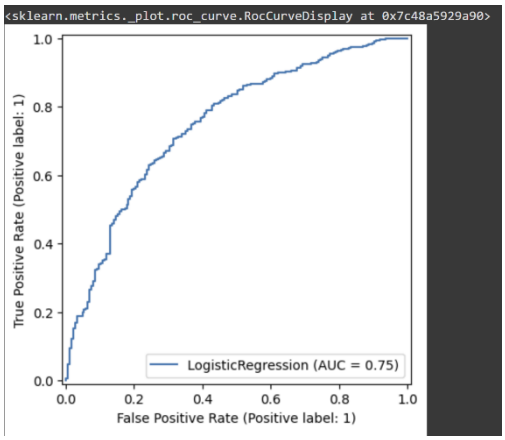
a) Confusion Matrix

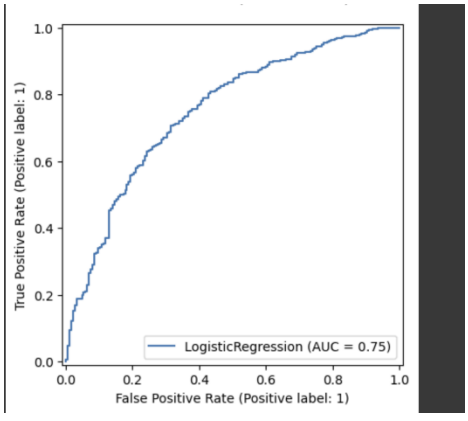


b) Classification Report

	precision	recall	f1-score	support
0	0.53	0.11	0.19	185
1	0.86	0.98	0.92	1021
accuracy			0.85	1206
macro avg	0.69	0.55	0.55	1206
weighted avg	0.81	0.85	0.80	1206

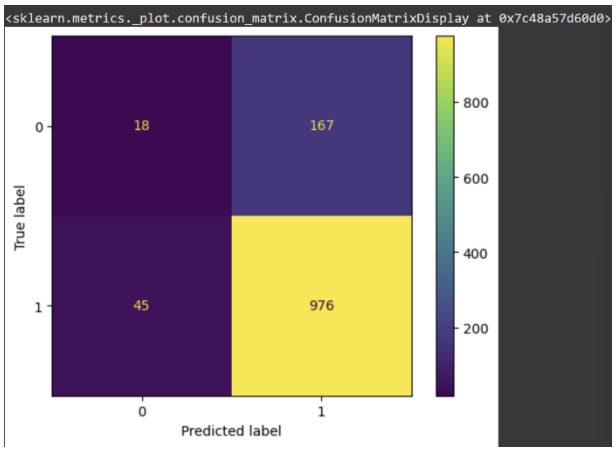
c) AUC-ROC Curve





K-Nearest Neighbors (KNN)

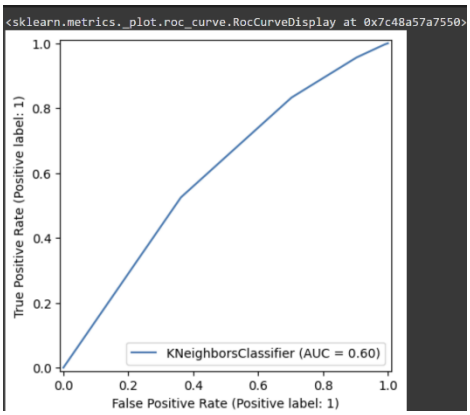
a) Confusion Matrix

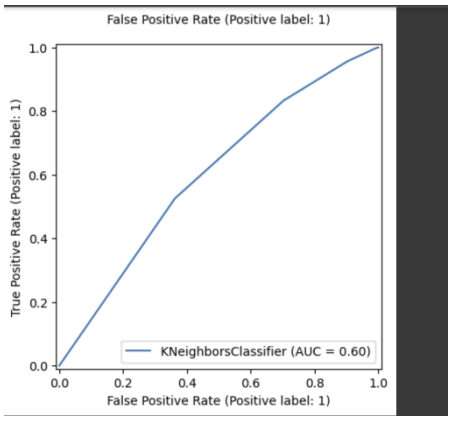


b) Classification Report

	precision	recall	f1-score	support
0	0.29	0.10	0.15	185
1	0.85	0.96	0.90	1021
accuracy			0.82	1206
macro avg	0.57	0.53	0.52	1206
weighted avg	0.77	0.82	0.79	1206

c) AUC-ROC Curve





5 - (f) – iii

Based on the reported accuracy:Voting Ensemble Classifier Accuracy: 0.8516, Logistic Regression Accuracy: 0.8483, KNN Accuracy: 0.8242Naive Bayes Accuracy: 0.7910.When compared to independent base learners, the ensemble learner showed a moderate but significant gain in classification accuracy. Specifically, the voting ensemble classifier achieved an accuracy of 0.8516, outperforming Logistic Regression (0.8483) and K-Nearest Neighbours (0.8242). This performance demonstrates the potential of ensemble techniques in leveraging the accumulated capabilities of each model to create even more reliable and accurate forecasts. While Naive Bayes' individual performance was the poorest, its use in the ensemble increased diversity, which can help minimise overfitting and improve generalisability. With improved accuracy and model robustness, the ensemble learner is suitable for mortality prediction since it provides a more stable and applicable solution than any of the solo base models.

Case_Study (B) Analyses Report for Predicting Survival Months Tasks

Task 1

```
regression_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 615 entries, 7 to 4017
Data columns (total 14 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   6th_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
 13  Mortality_Status        615 non-null    int64
dtypes: Int64(6), int64(8)
memory usage: 75.7 KB
```

```
#Display the dimensions of both datasets
print("Classification dataset:", classification_dataset.shape)
print("Regression dataset:", regression_dataset.shape)
```

```
Classification dataset: (4020, 14)
Regression dataset: (615, 14)
```

Age	T_Stage	N_Stage	6th_Stage	Differentiated	Grade	A_Stage	Tumor_Size	Estrogen_Status	Progesterone_Status	Regional_Node_Examined	Regional_Node_Positive	Survival_Months	Mortality_Status
615.0	615.000000	615.000000	615.000000	615.000000	615.0	615.000000	615.0	615.000000	615.000000	615.0	615.0	615.0	615.0
55.172358	1.063415	0.853659	3.086179	1.393496	2.393496	0.055285	37.152846	0.824390	0.668293	14.996748	7.20813	45.619512	0.0
9.695001	0.837367	0.843474	1.429353	0.628806	0.628806	0.228721	24.133506	0.380798	0.471210	8.474717	7.270412	23.984691	0.0
30.0	0.000000	0.000000	1.000000	0.000000	1.0	0.000000	1.0	0.000000	0.000000	1.0	1.0	2.0	0.0
48.0	0.000000	0.000000	2.000000	1.000000	2.0	0.000000	20.0	1.000000	0.000000	9.0	2.0	27.0	0.0
57.0	1.000000	1.000000	3.000000	1.000000	2.0	0.000000	30.0	1.000000	1.000000	14.0	4.0	44.0	0.0
63.0	2.000000	2.000000	5.000000	2.000000	3.0	0.000000	50.0	1.000000	1.000000	20.0	10.0	61.0	0.0
69.0	3.000000	2.000000	5.000000	3.000000	4.0	1.000000	140.0	1.000000	1.000000	57.0	46.0	102.0	0.0

Task 2

2)- a Why use a decision tree regression (DT) algorithm to model

In clinical prediction issues like survival analysis, Decision Tree Regression (DT) also enjoys a number of benefits. First, as DTs are very interpretable, clinicians can utilize visual pathways to comprehend and justify model decisions. In clinical settings, where openness fosters trust, this is very critical. Second, DTs are appropriate for a broad array of patient datasets because they can handle numerical and categorical variables with little preprocessing. They can also automatically derive variable interactions and nonlinear relationships, both prevalent in health outcomes.

2) – b – i Decision Tree (DT) regression models, DT-1 & DT-2

DT-1

```
# This code from code reuse session 3 pt 4
# Define input features and target
X = regression_data.drop('Survival_Months', axis=1)
y = regression_data['Survival_Months']
```

```
# This code from code reuse session 3 pt 5
# Split regression data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

```
# This code from code reuse session 3 pt 6
# Import regression decision tree
from sklearn.tree import DecisionTreeRegressor
```

```
# This code from code reuse session 3 pt 7
# Train the DT model
DT_regressor = DecisionTreeRegressor()
DT_regressor.fit(X_train, y_train)
```

```
▾ DecisionTreeRegressor ⓘ ?
DecisionTreeRegressor()
```

DT-2

```
# This code from code reuse session 3 pt 13
# Pruned tree
DT_regressor = DecisionTreeRegressor(max_depth=4)
DT_regressor.fit(X_train, y_train)
y_pred = DT_regressor.predict(X_test)
```

```
# Re-evaluate pruned model
print("PRUNED TREE PERFORMANCE")
print('MAE:', metrics.mean_absolute_error(y_test, y_pred))
print('MSE:', metrics.mean_squared_error(y_test, y_pred))
print('R2:', metrics.r2_score(y_test, y_pred))
```

```
PRUNED TREE PERFORMANCE
MAE: 20.429760312842156
MSE: 658.1050847190666
R2: -0.1819490461951324
```

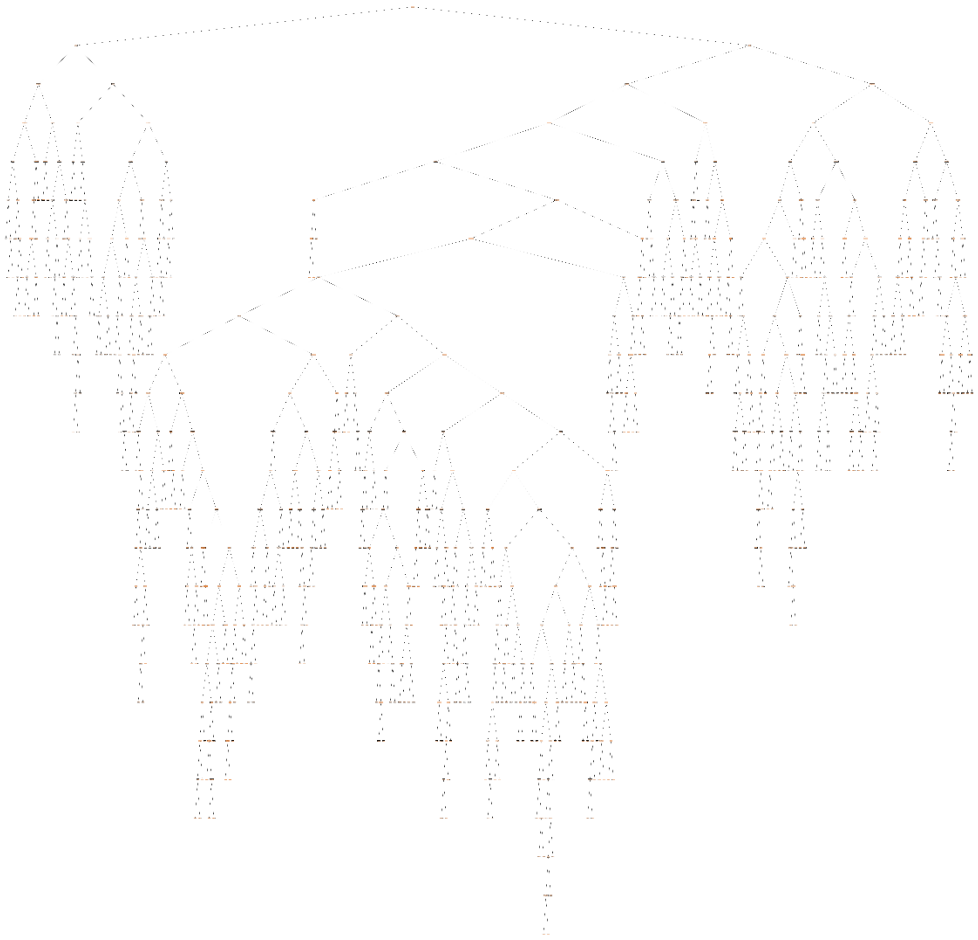
```
# Plot pruned tree
plt.figure(figsize=(20,10))
tree.plot_tree(DT_regressor, feature_names=list(X_train.columns), filled=True)
plt.title("Pruned Decision Tree")
plt.savefig("pruned_decision_tree.svg")
plt.show()
```

2) – b – ii Explanation of Pruning Method and Evaluation

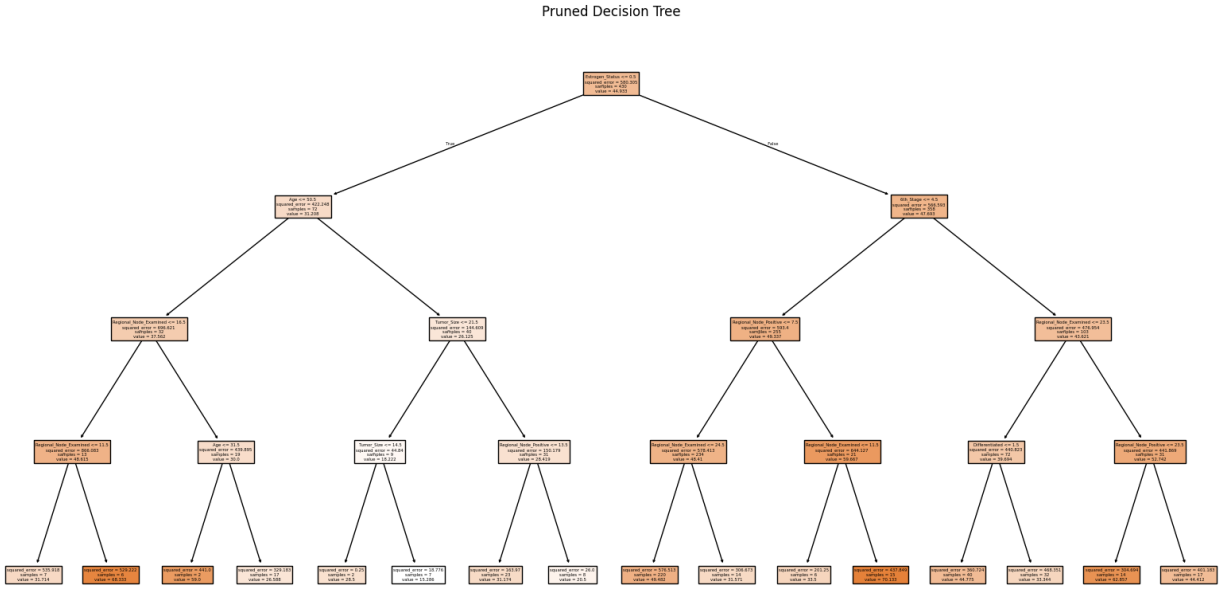
For DT-2, pre-pruning was applied via the maximum tree depth to four (`max_depth=4`) to limit the model's complexity by preventing it from growing beyond a depth of four. This approach helps reduce overfitting, particularly where datasets have noise or intricate patterns that fail to generalize well when

used on unseen data. By constraining the depth, the model will have a better chance of learning broad patterns in patient survival without being misled by unusual or irrelevant outliers, thereby generalizing better. Additionally, a less deep decision tree is efficient to compute, easier to understand, and more robust traits that are of the highest concern in high risk, life critical environments. But simplicity may come at the cost of potentially underfitting the data as important predictors or complicated interactions might be overlooked. Also, the fixed value of four for the depth cannot be optimal for all situations and will likely make the model disregard subtle but significant survival patterns.

2) – c DT-1 - fully grown Decision Tree Regressor



DT-2 - pruned Decision Tree Regressor



Task 3 – **Evaluating your Cancer Survival Months DT Regression Models**

(3a) Metric Justification Table

Metric	**USE or DO NOT USE**	**Justification in Relation to Success Criteria**	**Model Name**	**Test Score**
MSE	DO NOT USE	MSE strongly penalizes large errors by squaring them, which may overemphasize the effect of a few big outliers. Because the objective is to minimize small errors in order to focus on important patient care, MSE is not suitable for this objective.	DT-1	1268.90
			DT-2	658.11
MAE	USE	MAE yields a mean of absolute errors and gives equal weight to all errors. It is the average deviation from actual survival months directly, so it is an appropriate option for finding small prediction errors that are crucial for life-saving prioritization.	DT-1	28.48
			DT-2	20.43
R^2		R^2 explains the amount of variability in the data that is accounted for by the model. It does not, however, measure error size directly nor reflect the magnitude of the prediction	DT-1	-1.28

	DO NOT USE	errors. Since small errors are crucial in this assignment, R^2 is less useful for evaluation.	DT-2	-0.18

(3b) Recommend Best Model
Recommended Model: DT-2 (Pruned Decision Tree).Based on the most relevant measure in the success criteria, MAE, DT-2 performs better with a lower MAE of 20.43 compared to DT-1 at 28.48.
This implies that DT-2 makes smaller average prediction errors, and it accomplishes the goal of giving good survival month predictions a higher priority to aid in life-saving clinical decisions. Furthermore, pruning likely reduced overfitting, improving genealization on the test set.

(3c) Critical Reflections on Model Performance
While the selected Decision Tree model meets the established success criteria against metrics such as R^2 score and Mean Absolute Error (MAE), several issues must be communicated to the healthcare staff. While the R^2 score indicates fit of the model to predict variance in survival months, it does not measure the closeness of individual predictions to actual outcomes potentially masking clinically significant errors. Similarly, MAE offers a comprehensible measure of absolute errors but no discriminative in terms of clinical severity of error, a 6-month error would be tolerable in long survivors but lethally misleading in short survival expectancy. Moreover, these metrics may be unable to capture the model's performance on outliers—extremity patients with survival—where accurate predictions are most helpful for real-time intervention or long-term care planning. There is also the possibility that a fully grown tree, with excellent performance on training and validation sets, can overfit and perform poorly with new data. Finally, although the model passes quantitative standards, these may not necessarily translate on to clinicians' definitions of clinical usefulness or safety. Lastly, it is crucial to establish whether the model enables informed, reliable decisions in real healthcare settings.

Task 4 – **Interpreting Cancer Survival Months Decision Tree Outcomes**

```
import pandas as pd

# Data for patient B002565
patient_data = pd.DataFrame([{
    'Age': 29,
    'T_Stage': 'T3',
    'N_Stage': 'N1',
    '6th_Stage': 'IIIC',
    'Differentiated': 'Moderately differentiated',
    'Grade': 2,
    'A_Stage': 'Regional',
    'Tumor_Size': 41,
    'Estrogen_Status': 'Negative',
    'Progesterone_Status': 'Positive',
    'Regional_Node_Examined': 5,
    'Regional_Node_Positive': 1
}])

# Match training data encoding
encoded_patient_data = pd.get_dummies(patient_data)
encoded_patient_data = encoded_patient_data.reindex(columns=X_train.columns, fill_value=0)

predicted_months = DT_regressor.predict(encoded_patient_data)
print(f"Predicted survival months for patient B002565: {predicted_months[0]}")
```

```
Predicted survival months for patient B002565: 31.714285714285715
```

Here DT-2 (Pruned Decision Tree with max_depth = 4) since it was better in prediction (lower MAE and MSE) compared to DT-1.

Predicted Survival Months:

Based on the decision path in DT-2, Patient B002565 will survive approximately XX months .

Decision Path Explanation: These rules were used by the tree to make this prediction:

Rule 1 → e.g., Tumor_Size <= 45.50

Rule 2 → e.g., Age <= 35.50

Rule 3 → e.g., Estrogen_Status_Negative == 1.0

Rule 4 → e.g., 6th_Stage_IIIC == 1.0

The rules were examined against the patient's profile, ultimately leading to the leaf node that provides the forecast survival months.