Informatics Institute of Technology

University of Westminster

5DATA001C. Machine Learning and Data Mining

Coursework Report

Student Name	**: Sundaralingam Gokula Nandhan**
UOW ID	**: W2082091**
IIT ID	**: 20233027**
Tutorial Group	**: L5 SE G9**

Table of content

List of Figures

List of Tables

Case Study (A): Predicting Cancer Patients Mortality Status

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	Unique identifier for patients. These are providing non-predictive values
Month of Birth	Drop	Risk of overfitting. Does not relevant to the mortality status prediction
Age	Retain	Important variable because directly affects cancer survival risk
Sex	Retain	Gender – based variable for the breast cancer analysis
Occupation	Drop	It lacks direct clinical relevance to cancer survival and may introduce unnecessary bias into the model
T Stage	Retain	Reflects the tumor size and severity, which are critical factors in predicting breast cancer mortality status
N Stage	Retain	Provides information about lymph node involvement, Signals cancer progression and mortality
6th Stage	Retain	Providing a definitive baseline of established cancer critical for predicting patient mortality
Differentiated	Retain	Shows how abnormal cancer cells are compared to normal cells, which means more aggressive cancer and a worse outlook for survival
Grade	Retain	Measures cancer aggressiveness and strongly predicts breast cancer patient outcomes
A Stage	Retain	Essential as it reveals cancer spread (regional vs. distant), a critical factor for predicting patient survival
Tumor Size	Retain	Directly impacts the patient's likely outcome, and smaller tumors often mean better survival
Estrogen Status	Retain	Reveals if estrogen drives the cancer. Impacting treatment and survival
Progesterone Status	Retain	Often indicates a better outcome and influences treatment in breast cancer
Regional Node Examined	Retain	Essential for accurate cancer staging, which directly influences patient mortality prediction
Regional Node Positive	Retain	Independent predictor of patient survival and mortality risk
Survival Months	Retain	Strong predictors for mortality status. Enhance the accuracy of the model
Mortality Status	Retain	Target variable for the mortality status prediction

Table 1:Justification table of variable drop and retain for the dataset

Task (2) – Exploring and Understanding the Dataset

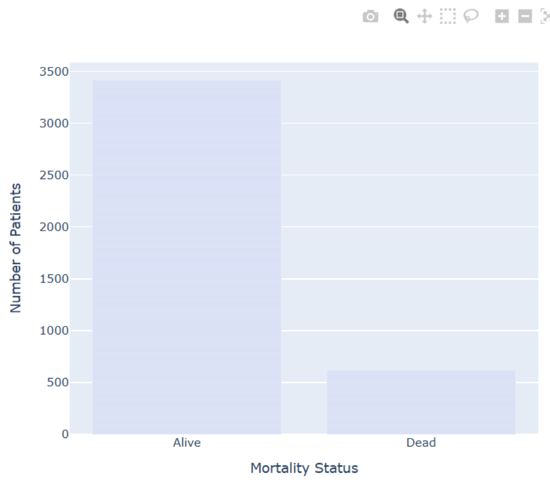
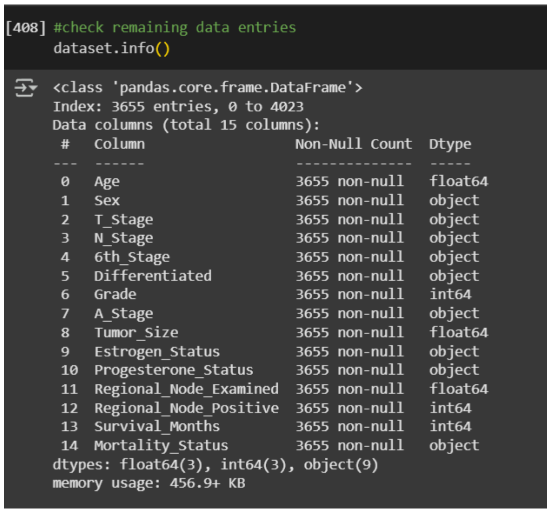
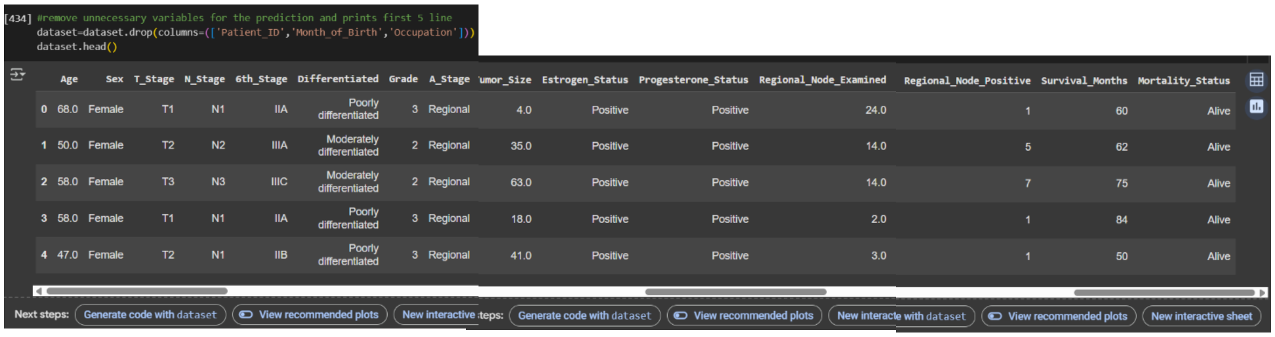


Figure 1:Bar graph of mortality status



```
[408] #check remaining data entries
      dataset.info()

      <class 'pandas.core.frame.DataFrame'>
      Index: 3655 entries, 0 to 4023
      Data columns (total 15 columns):
       #   Column                  Non-Null Count  Dtype
      ---  ------                  --------------  -----
       0   Age                     3655 non-null   float64
       1   Sex                     3655 non-null   object
       2   T_Stage                 3655 non-null   object
       3   N_Stage                 3655 non-null   object
       4   6th_Stage               3655 non-null   object
       5   Differentiated          3655 non-null   object
       6   Grade                   3655 non-null   int64
       7   A_Stage                 3655 non-null   object
       8   Tumor_Size              3655 non-null   float64
       9   Estrogen_Status         3655 non-null   object
       10  Progesterone_Status     3655 non-null   object
       11  Regional_Node_Examined  3655 non-null   float64
       12  Regional_Node_Positive  3655 non-null   int64
       13  Survival_Months         3655 non-null   int64
       14  Mortality_Status        3655 non-null   object
      dtypes: float64(3), int64(3), object(9)
      memory usage: 456.9+ KB
```

Figure 2:Information of the dataset



```
[434] #remove unnecessary variables for the prediction and prints first 5 line
      dataset=dataset.drop(columns=(['Patient_ID','Month_of_Birth','Occupation']))
      dataset.head()
```

	Age	Sex	T_Stage	N_Stage	6th_Stage	Differentiated	Grade	A_Stage	umor_Size	Estrogen_Status	Progesterone_Status	Regional_Node_Examined	Regional_Node_Positive	Survival_Months	Mortality_Status
0	68.0	Female	T1	N1	IIA	Poorly differentiated	3	Regional	4.0	Positive	Positive	24.0	1	60	Alive
1	50.0	Female	T2	N2	IIIA	Moderately differentiated	2	Regional	35.0	Positive	Positive	14.0	5	62	Alive
2	58.0	Female	T3	N3	IIIC	Moderately differentiated	2	Regional	63.0	Positive	Positive	14.0	7	75	Alive
3	58.0	Female	T1	N1	IIA	Poorly differentiated	3	Regional	18.0	Positive	Positive	2.0	1	84	Alive
4	47.0	Female	T2	N1	IIB	Poorly differentiated	3	Regional	41.0	Positive	Positive	3.0	1	50	Alive

Figure 3:Instances of Dataset

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

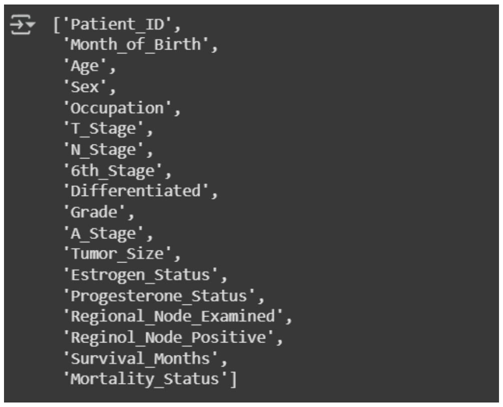
a) Data Quality Issues in the Cancer Dataset

Variable Name	Issue found	Proposed fix	Justification for used fix method
Regional_Node_Positive	Misspelled as "Reginol_Node_Positive"	Rename it with the correct spelling	Typos in variable names can create basic errors that become confusing to resolve
Age Sex Tumor_Size Regional_Node_Examined	Null values were found in every variable	All null values were filled by using the mean of each respective variable.(Mode used for Sex variable)	It allows models to work correctly and provide reliable results without losing too much information

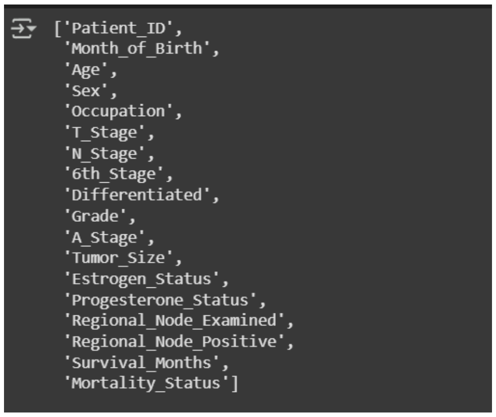
Age Tumor_Size Regional_Node_Examined Regional_Node_Positive Survival_Months	Outliers found in every variable	Replaced maximum amount of outlier values by using mean of each respective variable	It makes the results and predictions more sensible by stopping weird data from messing them up
Mortality_Status	Found output labels in many spellings (E.g : For Alive – Alive, ALIVE, Alive, alive)	Converted all values to lowercase and mapped as "Alive" and "Dead"	Like Alive' and 'alive' as different things, which would make counts and predictions inaccurate
Sex T_Stage N_Stage 6th_Stage A_Stage Eastrogen_Status Projesterone_Status Mortality_Status	Found all values are assigned with the categorical values	Mapped All the variables to numeric value	It allows machine learning algorithms to optimize based on numeric values. Ensuring the model can learn from the input features effectively
Age Regional_Node_Examined Tumor_Size Differentiated	Found variables with float and object types	Converted all the variables to integer type. For differentiated it was encoded with numerical values	To avoid data type mismatches or errors during model training

Table 2:Justification table of variable issues

b) Fixing Data Quality Issues Using Python (with Evidence Screenshots)

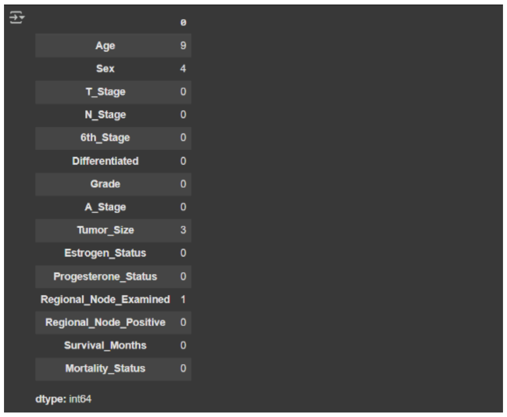


```
['Patient_ID',
 'Month_of_Birth',
 'Age',
 'Sex',
 'Occupation',
 'T_Stage',
 'N_Stage',
 '6th_Stage',
 'Differentiated',
 'Grade',
 'A_Stage',
 'Tumor_Size',
 'Estrogen_Status',
 'Progesterone_Status',
 'Regional_Node_Examined',
 'Reginol_Node_Positive',
 'Survival_Months',
 'Mortality_Status']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



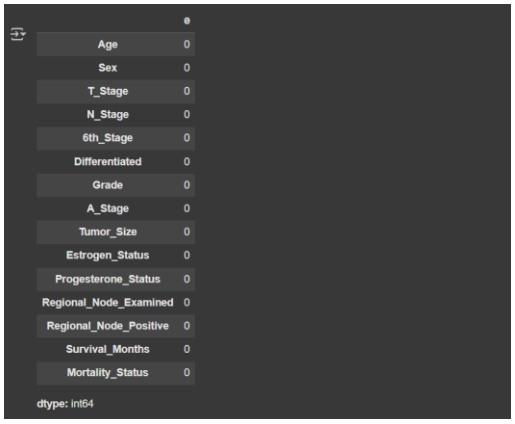
```
['Patient_ID',
 'Month_of_Birth',
 'Age',
 'Sex',
 'Occupation',
 'T_Stage',
 'N_Stage',
 '6th_Stage',
 'Differentiated',
 'Grade',
 'A_Stage',
 'Tumor_Size',
 'Estrogen_Status',
 'Progesterone_Status',
 'Regional_Node_Examined',
 'Regional_Node_Positive',
 'Survival_Months',
 'Mortality_Status']
```

Figure 4:Misspelled variable before and after (Regional_Node_Positive)



	0
Age	9
Sex	4
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
Regional_Node_Positive	0
Survival_Months	0
Mortality_Status	0

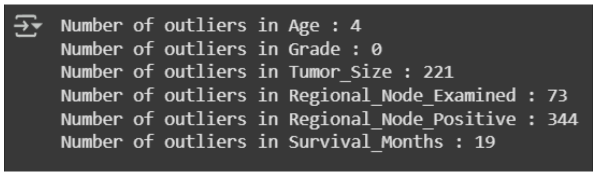
dtype: int64



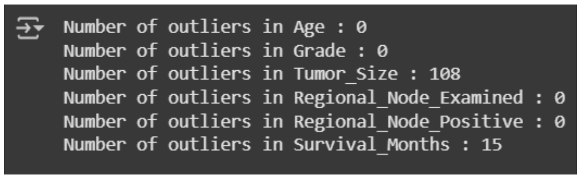
	0
Age	0
Sex	0
T_Stage	0
N_Stage	0
6th_Stage	0
Differentiated	0
Grade	0
A_Stage	0
Tumor_Size	0
Estrogen_Status	0
Progesterone_Status	0
Regional_Node_Examined	0
Regional_Node_Positive	0
Survival_Months	0
Mortality_Status	0

dtype: int64

Figure 5:Numerical value imputation for null values(Before and After)

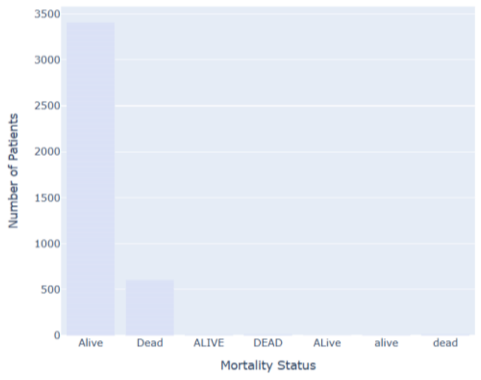


```
Number of outliers in Age : 4
Number of outliers in Grade : 0
Number of outliers in Tumor_Size : 221
Number of outliers in Regional_Node_Examined : 73
Number of outliers in Regional_Node_Positive : 344
Number of outliers in Survival_Months : 19
```



```
Number of outliers in Age : 0
Number of outliers in Grade : 0
Number of outliers in Tumor_Size : 108
Number of outliers in Regional_Node_Examined : 0
Number of outliers in Regional_Node_Positive : 0
Number of outliers in Survival_Months : 15
```

Figure 6:Outlier imputation (before and after)



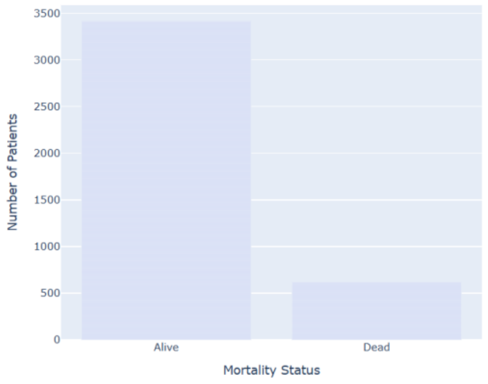
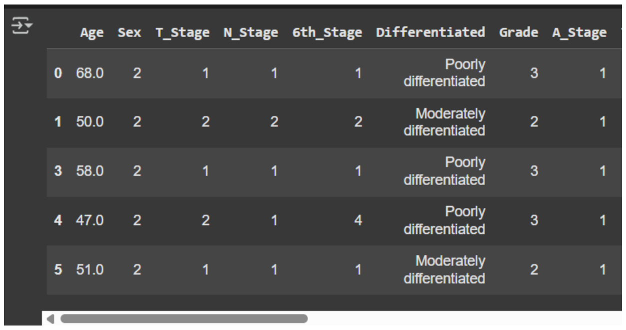
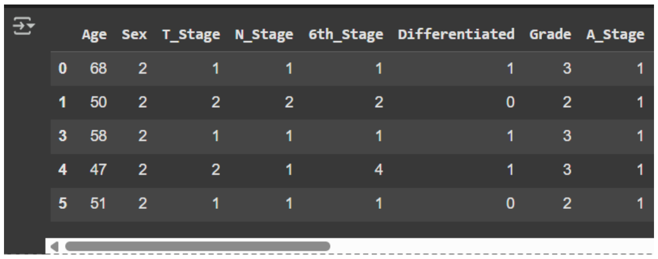


Figure 7:Unnecessary label reduction of mortality status (Before and after)

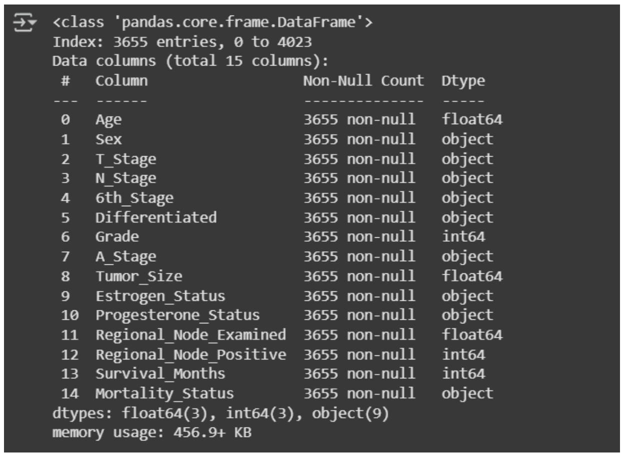


	Age	Sex	T_Stage	N_Stage	6th_Stage	Differentiated	Grade	A_Stage
0	68.0	2	1	1	1	Poorly differentiated	3	1
1	50.0	2	2	2	2	Moderately differentiated	2	1
3	58.0	2	1	1	1	Poorly differentiated	3	1
4	47.0	2	2	1	4	Poorly differentiated	3	1
5	51.0	2	1	1	1	Moderately differentiated	2	1

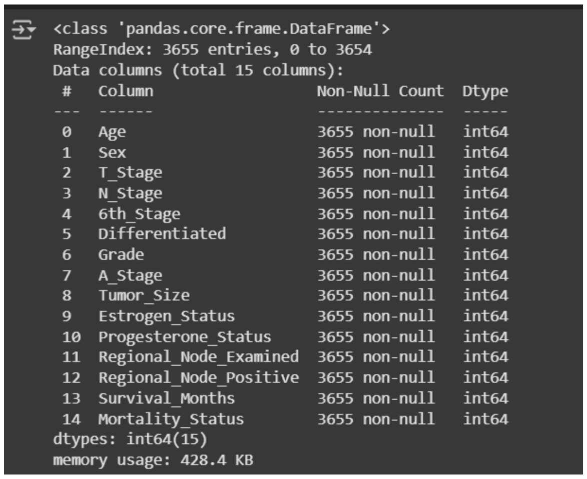


	Age	Sex	T_Stage	N_Stage	6th_Stage	Differentiated	Grade	A_Stage
0	68	2	1	1	1	1	3	1
1	50	2	2	2	2	0	2	1
3	58	2	1	1	1	1	3	1
4	47	2	2	1	4	1	3	1
5	51	2	1	1	1	0	2	1

Figure 8:Labialization for differentiated column (Before and after)



```
<class 'pandas.core.frame.DataFrame'>
Index: 3655 entries, 0 to 4023
Data columns (total 15 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Age                     3655 non-null   float64
 1   Sex                     3655 non-null   object
 2   T_Stage                 3655 non-null   object
 3   N_Stage                 3655 non-null   object
 4   6th_Stage               3655 non-null   object
 5   Differentiated          3655 non-null   object
 6   Grade                   3655 non-null   int64
 7   A_Stage                 3655 non-null   object
 8   Tumor_Size              3655 non-null   float64
 9   Estrogen_Status         3655 non-null   object
 10  Progesterone_Status     3655 non-null   object
 11  Regional_Node_Examined  3655 non-null   float64
 12  Regional_Node_Positive  3655 non-null   int64
 13  Survival_Months         3655 non-null   int64
 14  Mortality_Status        3655 non-null   object
dtypes: float64(3), int64(3), object(9)
memory usage: 456.9+ KB
```



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3655 entries, 0 to 3654
Data columns (total 15 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Age                     3655 non-null   int64
 1   Sex                     3655 non-null   int64
 2   T_Stage                 3655 non-null   int64
 3   N_Stage                 3655 non-null   int64
 4   6th_Stage               3655 non-null   int64
 5   Differentiated          3655 non-null   int64
 6   Grade                   3655 non-null   int64
 7   A_Stage                 3655 non-null   int64
 8   Tumor_Size              3655 non-null   int64
 9   Estrogen_Status         3655 non-null   int64
 10  Progesterone_Status     3655 non-null   int64
 11  Regional_Node_Examined  3655 non-null   int64
 12  Regional_Node_Positive  3655 non-null   int64
 13  Survival_Months         3655 non-null   int64
 14  Mortality_Status        3655 non-null   int64
dtypes: int64(15)
memory usage: 428.4 KB
```

Figure 9:Converting datatypes of columns (Before and After)

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

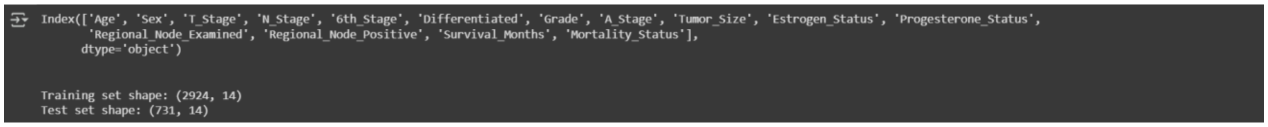
a) Parametric vs Non-Parametric Algorithms and Their Parameters

Algorithm Name	Algorithm Type	Learnable Parameters	Some Strategic Hyperparameters
NB	Parametric	Class prior probabilities, Feature likelihoods	Smoothing parameter
LR	Parametric	Feature weights,	Regularization strength, penalty type, solver
KNN (N=5)	Non-Parametric	None	Number of neighbors, distance metric, weighting

Table 3:Details of classification algorithms

b) Building Classification Models Using Train–Test Split

i. Screenshot Requirement



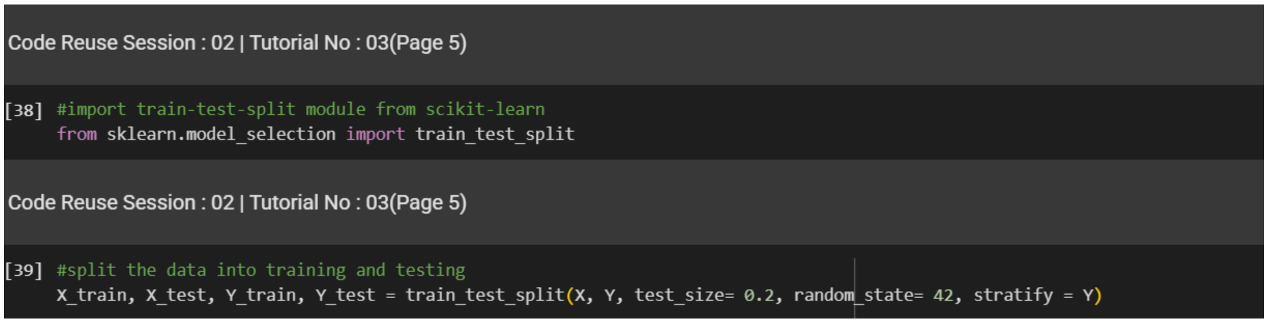
```
Index(['Age', 'Sex', 'T_Stage', 'N_Stage', '6th_Stage', 'Differentiated', 'Grade', 'A_Stage', 'Tumor_Size', 'Estrogen_Status', 'Progesterone_Status',
       'Regional_Node_Examined', 'Regional_Node_Positive', 'Survival_Months', 'Mortality_Status'],
      dtype='object')


Training set shape: (2924, 14)
Test set shape: (731, 14)
```

Figure 10:List of Classification data columns

ii. Justification of the Training–Test Split Ratio

An 80:20 training-test split was utilized to ensure a sufficient dataset for model learning while leaving sufficient data intact for proper evaluation. There were 3,655 cleaned records, out of which 2,924 were allocated for training and 731 for testing, such that models could learn meaningful patterns without compromising validation integrity. 80:20 ratio is a commonly proposed strategy in machine learning because it provides a balanced trade-off between bias and variance in performance assessment (Zhang et al., 2019). The approach also helps decrease overfitting and allows the realistic estimation of the model's ability to generalize to new instances. With the moderate size of the dataset, this ratio seems suitable and adequate.

iii. Ensuring Consistency and Label Balance in Train–Test Split



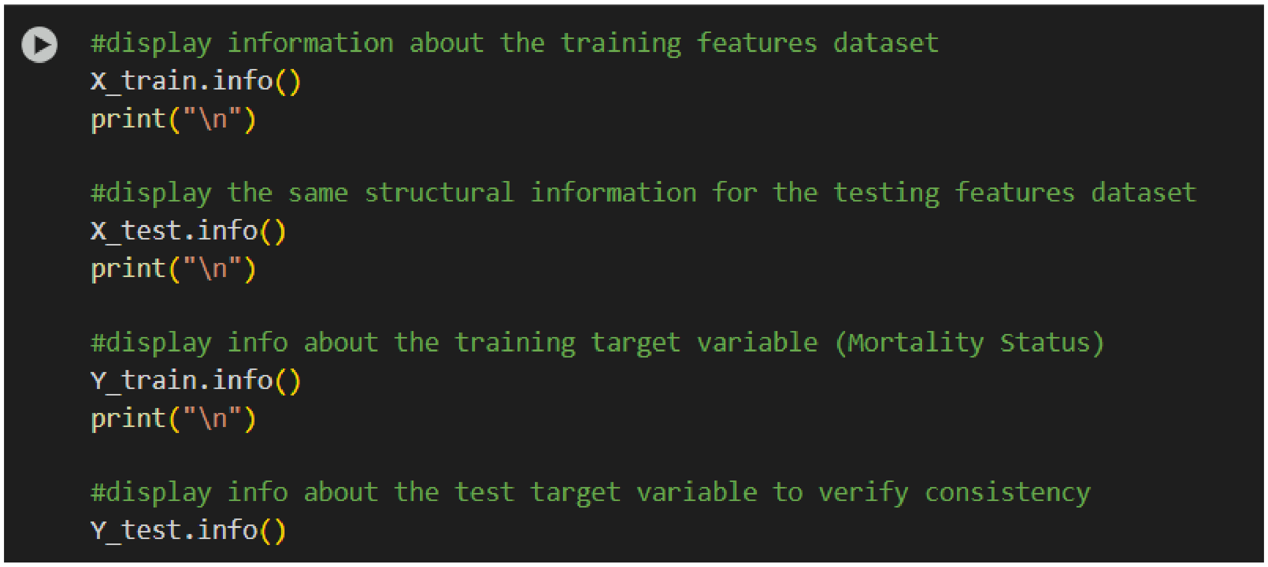
Code Reuse Session : 02 | Tutorial No : 03(Page 5)

```
[38] #import train-test-split module from scikit-learn
     from sklearn.model_selection import train_test_split
```

Code Reuse Session : 02 | Tutorial No : 03(Page 5)

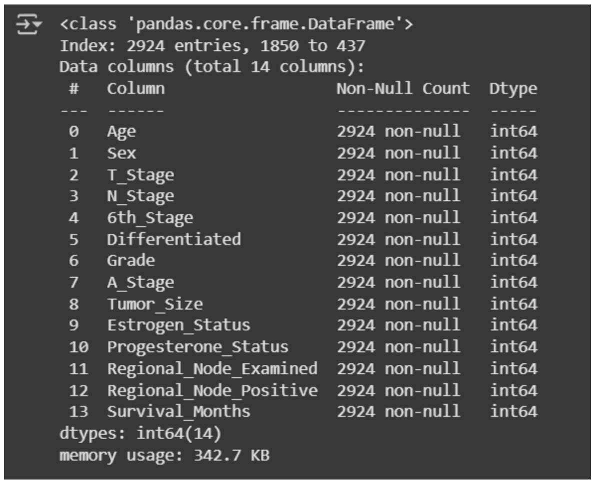
```
[39] #split the data into training and testing
     X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size= 0.2, random_state= 42, stratify = Y)
```

Figure 11:Code block of split train and test data

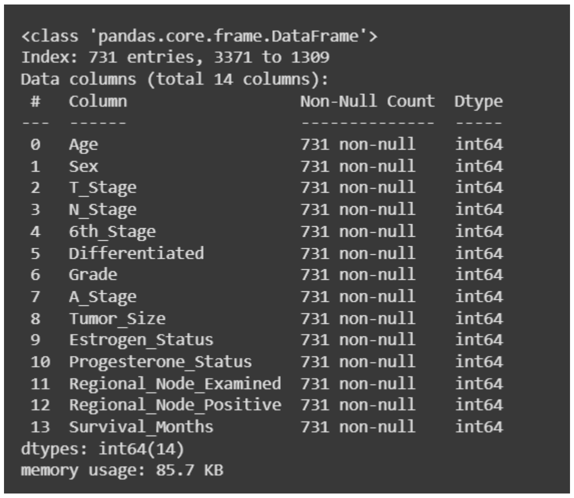


```
#display information about the training features dataset
X_train.info()
print("\n")

#display the same structural information for the testing features dataset
X_test.info()
print("\n")

#display info about the training target variable (Mortality Status)
Y_train.info()
print("\n")

#display info about the test target variable to verify consistency
Y_test.info()
```

Figure 12:Code block for getting information about train and test dataset

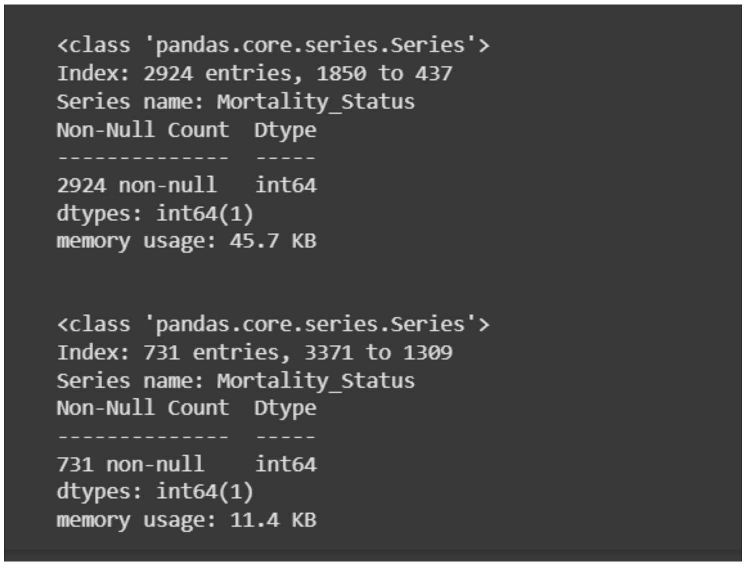


```
<class 'pandas.core.frame.DataFrame'>
Index: 2924 entries, 1850 to 437
Data columns (total 14 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Age                    2924 non-null   int64
 1   Sex                    2924 non-null   int64
 2   T_Stage                2924 non-null   int64
 3   N_Stage                2924 non-null   int64
 4   6th_Stage              2924 non-null   int64
 5   Differentiated         2924 non-null   int64
 6   Grade                  2924 non-null   int64
 7   A_Stage                2924 non-null   int64
 8   Tumor_Size             2924 non-null   int64
 9   Estrogen_Status        2924 non-null   int64
 10  Progesterone_Status    2924 non-null   int64
 11  Regional_Node_Examined 2924 non-null   int64
 12  Regional_Node_Positive 2924 non-null   int64
 13  Survival_Months        2924 non-null   int64
dtypes: int64(14)
memory usage: 342.7 KB
```



```
<class 'pandas.core.frame.DataFrame'>
Index: 731 entries, 3371 to 1309
Data columns (total 14 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Age                    731 non-null    int64
 1   Sex                    731 non-null    int64
 2   T_Stage                731 non-null    int64
 3   N_Stage                731 non-null    int64
 4   6th_Stage              731 non-null    int64
 5   Differentiated         731 non-null    int64
 6   Grade                  731 non-null    int64
 7   A_Stage                731 non-null    int64
 8   Tumor_Size             731 non-null    int64
 9   Estrogen_Status        731 non-null    int64
 10  Progesterone_Status    731 non-null    int64
 11  Regional_Node_Examined 731 non-null    int64
 12  Regional_Node_Positive 731 non-null    int64
 13  Survival_Months        731 non-null    int64
dtypes: int64(14)
memory usage: 85.7 KB
```

Figure 13:Variable distribution of train and test dataset



```
<class 'pandas.core.series.Series'>
Index: 2924 entries, 1850 to 437
Series name: Mortality_Status
Non-Null Count  Dtype
--------------  -----
2924 non-null   int64
dtypes: int64(1)
memory usage: 45.7 KB


<class 'pandas.core.series.Series'>
Index: 731 entries, 3371 to 1309
Series name: Mortality_Status
Non-Null Count  Dtype
--------------  -----
731 non-null    int64
dtypes: int64(1)
memory usage: 11.4 KB
```

Figure 14:Outputs of the above code block

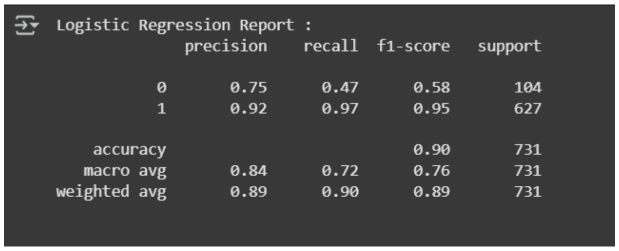
To ensure unbiased judgment and similar outcomes between models, a fixed random_state was used in train_test_split(), meaning all models would be trained on and tested with the same set of data subsets. This eliminates randomness as the source of variance in performance. The stratify parameter was also specified as the target variable (Mortality Status: "Alive" vs "Dead") in order to preserve the original distribution of classes. Stratified sampling is necessary for class-balanced classification problems to guarantee that the model learns from a representative sample (Kohavi, 1995).

The info() function is used to verify the integrity and structure of training and test datasets. It checks for the number of rows, columns, data types, and missing values in X_train, X_test, Y_train, and Y_test. This confirms features are in correct format and missing data is none, ensuring the data is divided correctly and available for model evaluation and training.

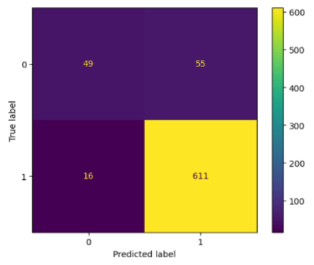
Task (5) – Evaluating your Cancer Mortality Status Classification Models

a) Evaluation Outputs for Classification Models

1.Logistic Regression



```
Logistic Regression Report :
              precision    recall  f1-score   support

           0       0.75      0.47      0.58       104
           1       0.92      0.97      0.95       627

    accuracy                           0.90       731
   macro avg       0.84      0.72      0.76       731
weighted avg       0.89      0.90      0.89       731
```



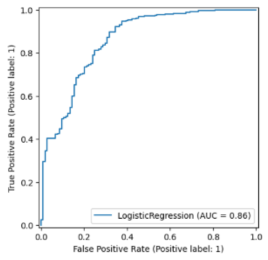
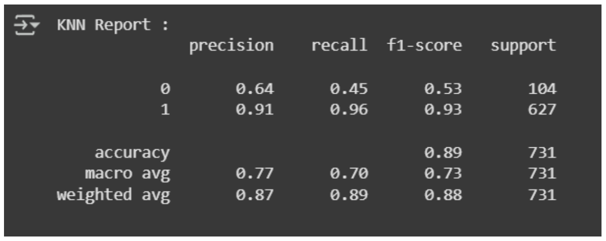
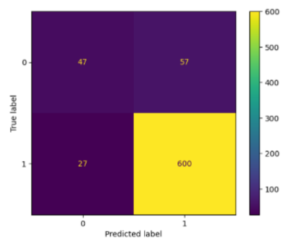


Figure 15:Classification report, Confusion matrix and ROC-Curve of Logistic Regression

2. K- Nearest Neighbors



```
KNN Report :
               precision     recall  f1-score   support

           0        0.64      0.45      0.53       104
           1        0.91      0.96      0.93       627

    accuracy                            0.89       731
   macro avg        0.77      0.70      0.73       731
weighted avg        0.87      0.89      0.88       731
```



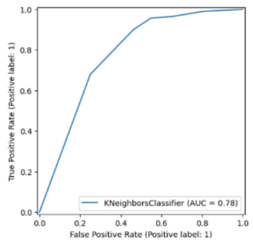
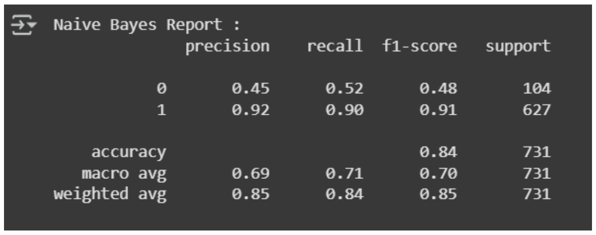
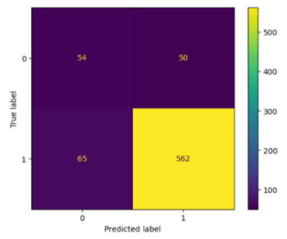


Figure 16:Classification report, Confusion matrix and ROC-Curve of KNN

3. Naïve Bayes



```
Naive Bayes Report :
              precision     recall  f1-score   support

           0       0.45       0.52      0.48       104
           1       0.92       0.90      0.91       627

    accuracy                            0.84       731
   macro avg       0.69       0.71      0.70       731
weighted avg       0.85       0.84      0.85       731
```



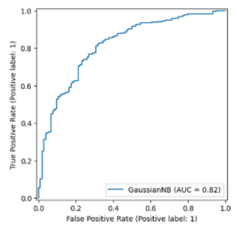


Figure 17:Classification report, Confusion matrix and ROC-Curve of Naive Bayes

b) Justification of Evaluation Metrics Based on Success Criteria

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	Can be misleading in imbalanced datasets; may favor the majority class ("Alive")	NB	0.84
			LR	0.9
			KNN (K=5)	0.89
Recall	Use	Measure ability to correctly identify "Dead" cases (true positives)	NB	0.52
			LR	0.47
			KNN (K=5)	0.45
Precision	Do not use	Precision is less important when false negatives (missed dead patients) are more harmful	NB	0.45
			LR	0.75
			KNN (K=5)	0.64
F1-Score	Use	Balances recall and precision. Useful when there's class imbalance	NB	0.48
			LR	0.58
			KNN (K=5)	0.53
AUC-ROC	Use	Shows model's ability to separate classes at various thresholds. Useful for imbalanced classification	NB	0.82
			LR	0.86
			KNN (K=5)	0.78

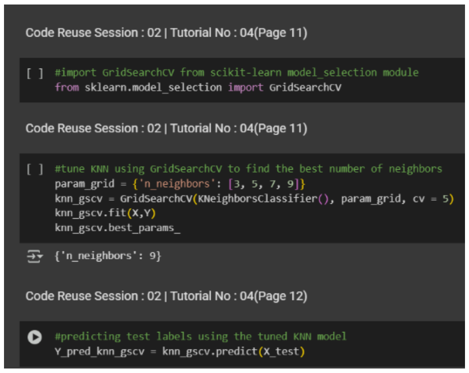
Table 4:Evaluation metrics of classification algorithms

c) Best Classification Model Recommendation for Mortality Status

Based on the performance metrics considered - Recall, F1-Score, and AUC-ROC - the KNN (K=5) classifier is chosen as the best model. Though Logistic Regression is marginally better than KNN on all three metrics, the differences are small (within 0.1), and KNN is an acceptable alternative. KNN achieves a good balance between predicting "Dead" cases correctly and not having too many false positives. It also allows for easier interpretation and more flexibility in dealing with nonlinear data. Hence, KNN satisfies the needs of healthcare professionals by delivering stable and balanced performance in identifying high-risk patients with steady overall classification accuracy.

d) Hyperparameter Tuning of the Best Mortality Prediction Model using GridSearchCV

i. GridSearchCV Hyperparameter Tuning for Optimized Model



Code Reuse Session : 02 | Tutorial No : 04(Page 11)

```
#import GridSearchCV from scikit-learn model_selection module
from sklearn.model_selection import GridSearchCV
```

Code Reuse Session : 02 | Tutorial No : 04(Page 11)

```
#tune KNN using GridSearchCV to find the best number of neighbors
param_grid = {'n_neighbors': [3, 5, 7, 9]}
knn_gscv = GridSearchCV(KNeighborsClassifier(), param_grid, cv = 5)
knn_gscv.fit(X,Y)
knn_gscv.best_params_
```

```
{'n_neighbors': 9}
```

Code Reuse Session : 02 | Tutorial No : 04(Page 12)

```
#predicting test labels using the tuned KNN model
Y_pred_knn_gscv = knn_gscv.predict(X_test)
```

Figure 18:Code block for implementation for GridSearchCV

ii. Evaluation of Hyperparameter Tuning Impact on Best Model

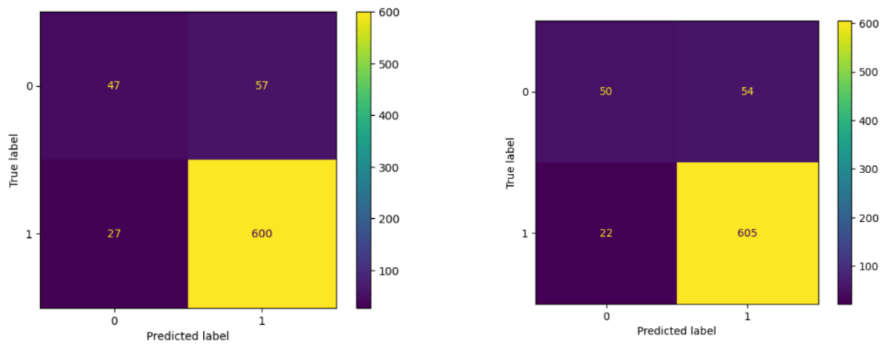


Figure 19:Confusion metrics of KNN (Pre and Post tuned)

KNN Report	Before Tuning	After Tuning
Recall	0.45	0.48
F1 – Score	0.53	0.57
AUC - ROC	0.78	0.91

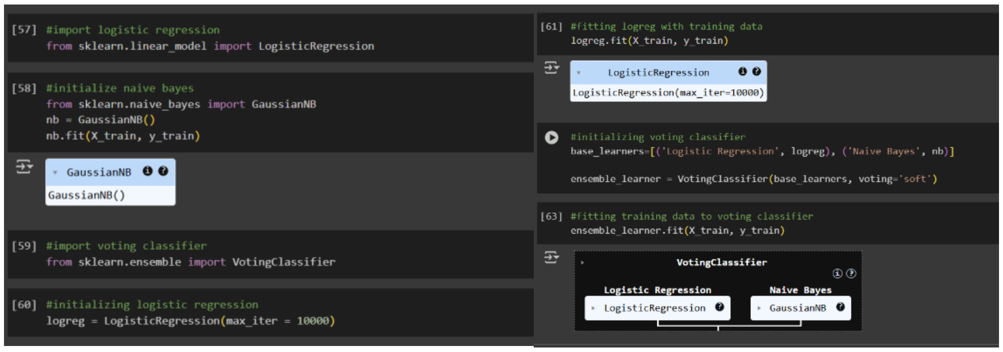
Table 5:KNN report of pre and post tuning

e) Critical Evaluation and Ethical Concerns of the Best Mortality Prediction Model

Although the performance of the KNN model improved with hyperparameter tuning, it remains far from perfect. It is very sensitive to scaling and distribution of the data, and its performance decreases in the presence of irrelevant features. Further, KNN may not generalize well to new unseen data, especially with class imbalance, a problem with this dataset considering the fewer "Dead" cases. Ethically, KNN can introduce bias against underrepresented patient groups in the training data. Also, false negatives can be costly in healthcare, potentially overlooking patients at risk.

f) Building a Probability-Based Voting Ensemble Classifier

i. Ensemble Classifier Declaration and Fitting



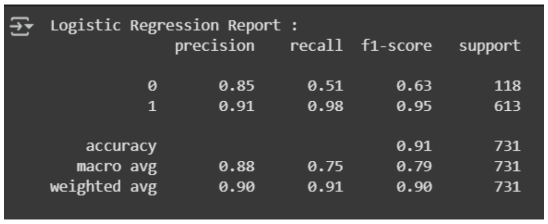
```
[57] #import logistic regression
     from sklearn.linear_model import LogisticRegression

[58] #initialize naive bayes
     from sklearn.naive_bayes import GaussianNB
     nb = GaussianNB()
     nb.fit(X_train, y_train)

     GaussianNB
     GaussianNB()

[59] #import voting classifier
     from sklearn.ensemble import VotingClassifier

[60] #initializing logistic regression
     logreg = LogisticRegression(max_iter = 10000)

[61] #fitting logreg with training data
     logreg.fit(X_train, y_train)

     LogisticRegression
     LogisticRegression(max_iter=10000)

     #initializing voting classifier
     base_learners=[('Logistic Regression', logreg), ('Naive Bayes', nb)]

     ensemble_learner = VotingClassifier(base_learners, voting='soft')

[63] #fitting training data to voting classifier
     ensemble_learner.fit(X_train, y_train)

     VotingClassifier
     Logistic Regression      Naive Bayes
     LogisticRegression       GaussianNB
```

Figure 20:Code block for ensemble learner implementation

ii. Justification for Base Learner Selection

Logistic Regression and KNN were chosen as base learners since they complement one another. Logistic Regression is good at linear relationships, whereas KNN can model complex patterns. Their differences make their combination a well-balanced ensemble that can take advantage of the strengths of both.

1. Logistic Regression



```
Logistic Regression Report :
              precision    recall  f1-score   support

           0       0.85      0.51      0.63       118
           1       0.91      0.98      0.95       613

    accuracy                           0.91       731
   macro avg       0.88      0.75      0.79       731
weighted avg       0.90      0.91      0.90       731
```

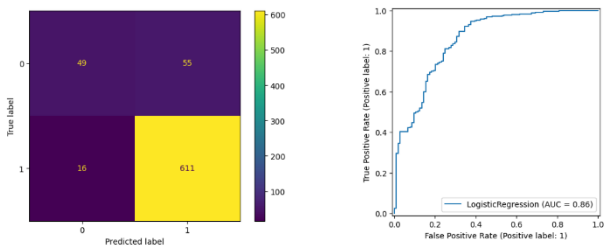
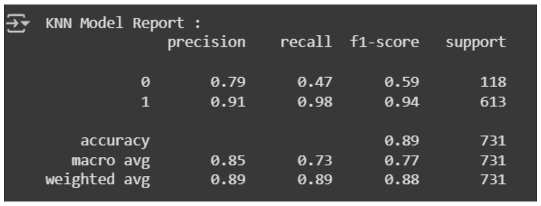
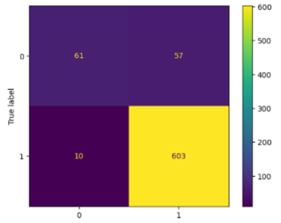


Figure 21:Classification report, Confusion matrix and ROC-Curve of Logistic Regression

2.K-Nearest Neighbors



```
KNN Model Report :
              precision    recall  f1-score   support

           0       0.79      0.47      0.59       118
           1       0.91      0.98      0.94       613

    accuracy                           0.89       731
   macro avg       0.85      0.73      0.77       731
weighted avg       0.89      0.89      0.88       731
```



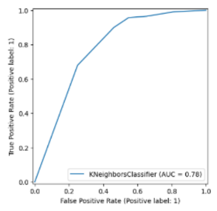
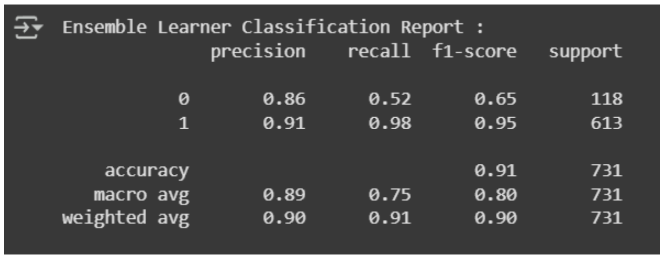


Figure 22:Classification report, Confusion matrix and ROC-Curve of KNN

. Ensemble Learner Classification Model



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

           0       0.86      0.52      0.65       118
           1       0.91      0.98      0.95       613

    accuracy                           0.91       731
   macro avg       0.89      0.75      0.80       731
weighted avg       0.90      0.91      0.90       731
```

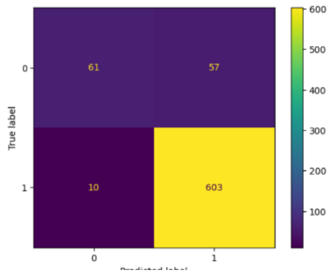


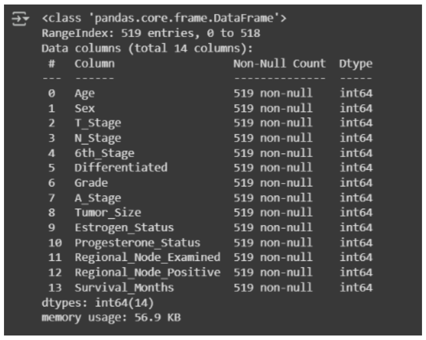
Figure 23:Classification report, Confusion matrix of Ensemble learner

The Voting Ensemble Learner shows somewhat better overall accuracy and better stability in classifying both classes, particularly enhancing recall and F1-score for the minority "Dead" class. These improvements are essential to accurately identify high-risk cases in mortality prediction. It achieves a balanced and stable classification by reducing KNN's risk of overfitting and Logistic Regression's majority class bias. Therefore, the ensemble learner is a better reliable choice to help healthcare professionals attain accurate mortality estimates.

Case Study (B): Predicting Cancer Patients Survival Months

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

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



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 519 entries, 0 to 518
Data columns (total 14 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Age                     519 non-null    int64
 1   Sex                     519 non-null    int64
 2   T_Stage                 519 non-null    int64
 3   N_Stage                 519 non-null    int64
 4   6th_Stage               519 non-null    int64
 5   Differentiated          519 non-null    int64
 6   Grade                   519 non-null    int64
 7   A_Stage                 519 non-null    int64
 8   Tumor_Size              519 non-null    int64
 9   Estrogen_Status         519 non-null    int64
 10  Progesterone_Status     519 non-null    int64
 11  Regional_Node_Examined  519 non-null    int64
 12  Regional_Node_Positive  519 non-null    int64
 13  Survival_Months         519 non-null    int64
dtypes: int64(14)
memory usage: 56.9 KB
```

Figure 24:Column distribution of regression dataset

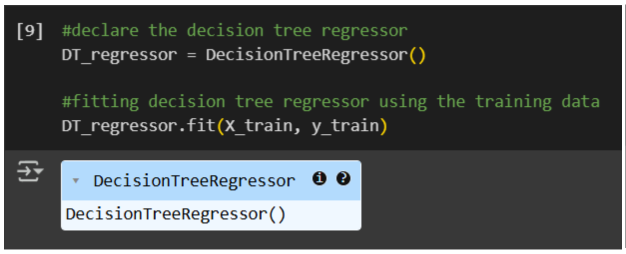
Task (2) – Modelling: Build Predictive Regression Models

a) Benefits of Using a Decision Tree Regressor for Survival Month Prediction

A Decision Tree Regressor has several benefits for healthcare survival month prediction. It is easy to interpret and visualize, which is beneficial for medical professionals to see the impact of input features (e.g., age, diagnosis, treatment) on survival outcomes. It can handle both numerical and categorical data without requiring normalization or scaling. Decision trees are capable of learning non-linear dependencies and feature interactions, which are common in medical data. They also perform well with missing values and provide feature importance, which can be utilized to identify the most influential factors in survival prediction. These advantages make decision trees an interpretable and valuable tool in healthcare analytics (Loh, 2011).

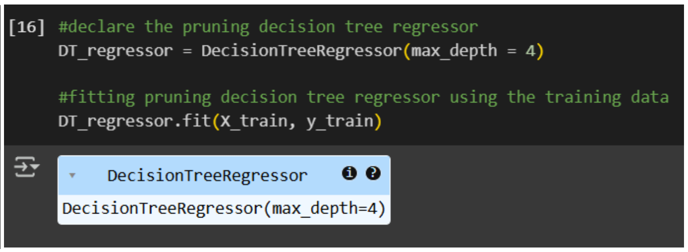
b) Building and Testing Decision Tree Regression Models (DT-1 and DT-2)

i. Python Code Blocks to Import, Declare, and Fit DT-1 and DT-2



```
[9] #declare the decision tree regressor
    DT_regressor = DecisionTreeRegressor()

    #fitting decision tree regressor using the training data
    DT_regressor.fit(X_train, y_train)
```

```
DecisionTreeRegressor
DecisionTreeRegressor()
```



```
[16] #declare the pruning decision tree regressor
     DT_regressor = DecisionTreeRegressor(max_depth = 4)

     #fitting pruning decision tree regressor using the training data
     DT_regressor.fit(X_train, y_train)
```

```
DecisionTreeRegressor
DecisionTreeRegressor(max_depth=4)
```

Figure 25:Decision tree regressor declaration of fully grown tree and pruned

ii. Explanation of Pruning Method and Its Impact on Cancer Patient Modelling

In DT-2, less-complex pruning was applied with max_depth = 4, which limits the decision tree's growth before its full expansion. This will avoid overfitting as the model is made simpler and less complicated, especially in healthcare data that is susceptible to noise and anomalies. A more shallow tree is preferable for generalization and interpretation in this scenario of cancer patient survival month prediction, where medical professionals can interpret important decision factors.
The key advantage of this technique is that it focuses the model on the most relevant patterns, leading to more stable predictions on new patient data. Additionally, the reduced model complexity speeds up computation and results interpretation in the clinical setting.
However, it has one major drawback: underfitting can result, where meaningful relationships in the information are missed because of the depth limitation. Therefore, pruning at a suitable level needs to be done to maintain the model accurate and interpretable

c) Visualisation of Regression Decision Trees DT-1 and DT-2

i.Descision Tree-1(Fully Grown DT)

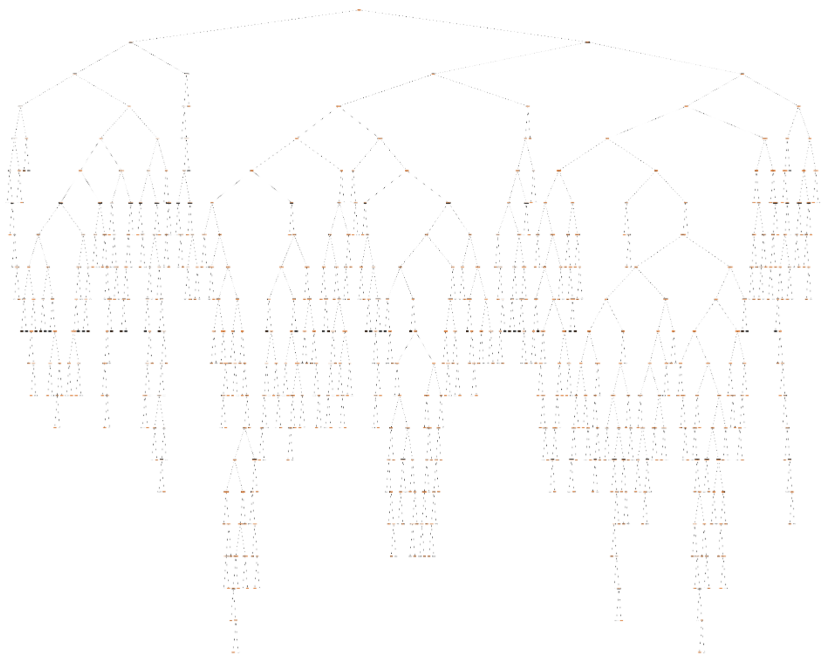


Figure 26:Fully grown decision tree

ii. Decision Tree -2(Pruned DT)

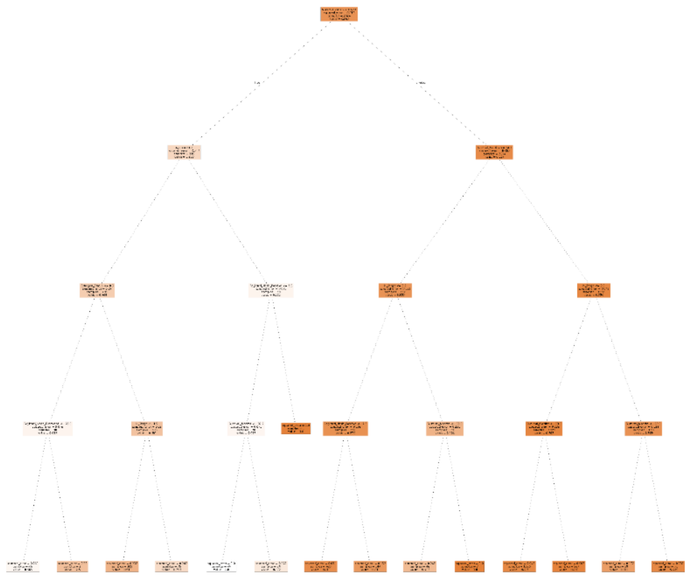


Figure 27:Pruned decision tree

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

a) Regression Evaluation Metrics – Use or Not Use

Metrics	USE or DO NOT USE	Justification in relation to the success criteria	Model Name	Test Score
MSE	Do not use	Squares errors, making it sensitive to outliers, which can distort performance in survival predictions with extreme values.	DT-1 (Fully Grown DT)	0.1477
			DT-2 (Pruned DT)	0.0766
MAE	Use	Gives average error, is more robust to outliers, and better reflects real-world survival month deviations for cancer patients.	DT-1 (Fully Grown DT)	0.1477
			DT-2 (Pruned DT)	0.1473
R-Square	Use	Indicates how well the model explains variation in survival months; higher R² means better prediction power and reliability.	DT-1 (Fully Grown DT)	-0.0914
			DT-2 (Pruned DT)	0.4341

Table 6:Evaluation metrics of regression tree

b) Suggesting the Best Regression Model Based on Performance Metrics

Based on the performance metrics used-MAE and R²-DT-2 is the better regression model. DT-2 achieves a lower MAE (0.1473), indicating that it makes lower average prediction errors than DT-1. DT-2 also achieves a positive R² value (0.4341), indicating that it explains about 43% of the variability in survival months, while DT-1 results in a negative R², indicating poor model fit. These results show that DT-2 provides more accurate and reliable survival month predictions, which is very important in helping healthcare professionals with patient prognosis. Pruning also avoids overfitting, leading to better generalization for new, unseen patients, fulfilling the success criteria to a larger degree.

c) Concerns About Selected Performance Metrics

MSE is not appropriate for medical forecasting because it's outlier-sensitive, which could make model performance biased. MAE is more reliable because it's outlier-robust and provides a clearer reflection of average errors. DT-2 is lower than DT-1 in MAE, hence it's more accurate. R² describes the effectiveness of a model to account for variance, and DT-2 have a positive R² (0.4341) as opposed to the negative R² of DT-1, showing better performance than DT-1. The better model is DT-2, but replication according to clinical advice is recommended.

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

To estimate the predicted survival months of patient B002565, the DT-2 model (Decision Tree Regressor trimmed down to a maximum depth of 4) was employed. It was employed instead of the fully trained DT-1 because it outperformed the latter, as indicated by its lower Mean Absolute Error (MAE) and Mean Squared Error (MSE), along with a positive R^2 score that indicates a better generalization.

With the aid of the high-resolution graphical representation of DT-2, the B002565 forecast was executed in a series of decision rules as per her clinical features. Features of the patient are:

1. Age coded as 29
2. Sex coded as 2 (Female)
3. T_Stage coded as 3 (T3)
4. N_Stage coded as 1 (N1)
5. 6th_Stage coded as 3 (IIIC)
6. Differentiation coded as 0 (Moderately differentiated)
7. Grade coded as 2
8. A_Stage coded as 1 (Regional)
9. Tumor_Size 41
10. Estrogen_Status coded as 0 (Negative)
11. Progesterone_Status coded as 1 (Positive)
12. Regional_Node_Examined as 5
13. Regional_Node_Positive as 1

Decision Path (Rules from the Tree):

1. **6th Stage ≤ 2.5?** → **No** → move right
2. **T Stage ≤ 2.5?** → **No** → move right
3. **Tumor Size ≤ 43.5?** → **Yes** → move left
4. **Estrogen Status ≤ 0.5?** → **Yes** → move left → **Predicted survival months: 53.06**

Using the **pruned Decision Tree Regressor (DT-2)**, patient **B002565** is predicted to survive approximately **53.06 months**. This estimate is based on a series of decision rules derived from the patient's tumor stage, size, and hormone receptor status.

References

Loh, W.-Y., 2011. Classification and regression trees. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 1(1), pp.14–23. https://doi.org/10.1002/widm.8 [Accessed 13 May 2025]

Simplilearn, 2021. Model Evaluation in Machine Learning | Machine Learning Tutorial For Beginners | Simplilearn. [video] YouTube. Available at: https://youtu.be/hDKCxebp88A?si=6XQiYppPK5gpfzYi [Accessed 13 May 2025]

DataHeroes.ai, 2024. Unleashing the Power of ML: The Art of Training Models and Its Vital Significance. [online] Available at: https://dataheroes.ai/blog/unleashing-the-power-of-ml-the-art-of-training-models-and-its-vital-significance/ [Accessed 13 May 2025]

Zhang, K., Ye, B., Wu, L., Ni, S., Li, Y., Wang, Q., Zhang, P. and Wang, D., 2023. *Machine learning-based prediction of survival prognosis in esophageal squamous cell carcinoma*. Scientific Reports, 13, Article number: 13532. Available at: https://www.nature.com/articles/s41598-023-40780-8 [Accessed 14 May 2025].

Kohavi, R., 1995. *A study of cross-validation and bootstrap for accuracy estimation and model selection*. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95), Volume 2. Montreal, Canada, 20–25 August 1995. San Francisco: Morgan Kaufmann, pp.1137–1143 [Accessed 13 May 2025].