**Assessment 1**

**Data Mining and Machine Learning (7BUIS008W.1)**

**Donald Philp | 21075797**

**2024**

**☕️ 😴**

**PART A:** **Breast Cancer Mortality Prediction**

1. **Domain Understanding**

**Task 1: Classification of Variables**

***Table****: Include a table of retained and dropped variables with justification.*

|  |  |  |
| --- | --- | --- |
| Variable Name | Retain or Drop | Brief Justification of Retention or Dropping |
| Patient ID | Drop | There is no predictive value in the identification of patients. |
| Month of Birth | Drop | Age is already provided. Therefore, fractions of a year in months and, more specifically, days do not add additional precision to our target output. |
| Age | Retain | Age is possibly a known factor in disease progression and survival rates. |
| Sex | Drop | In this data set, there is only one categorical value in the ‘sex' category: ' female'. |
| Race | Retain | Ethnicity is an important value that could influence outcome risk. |
| Marital Status | \*\*Retain | Social support, such as marital status, could impact survival rate and, therefore, some outcome on our target variable.  \*\* We would like to test this variable with and without it to confirm |
| Occupation Code | Drop | The occupancy code, relating to the specific occupancy type, has no direct relevance to cancer progression or outcome. |
| Adopted Status | Drop | The 'Adopted status' has no direct relevancy to cancer progression or outcome. |
| T Stage | Retain | Indicates tumour size in different stages; this would be critical to see survival sizes. |
| N Stage | Retain | This reflects the spread at specific stages in the lymph nodes, which are critical for severity and aggressiveness testing. |
| 6th Stage | Retain | This could play an essential role in understanding and predicting end-stage cancer lifecycle. |
| Differentiate | Retain | Differentiate is the sign of the aggressiveness of the cancer, which is very important to the outcome. |
| Grade | Retain | The cancer grade would be directly proportional to the outcome of the life or death. |
| A Stage | Retain | Indicates cancer spread, critical for diagnosing and understanding the severity of the cancer |
| Tumour Size | Retain | Tumour size would directly predict survival as the increased size could singly increase survival months. |
| Estrogen Status | Retain | Hormone receptor status informs possibly the spread of the cancer to some point. |
| Progesterone Status | Retain | Hormone receptor status informs possibly the spread of the cancer to some point. |
| Regional Node Examined | Retain | Adds context to lymph involvement and if the cancer would be in the lymph nodes. |
| Regional Node Positive | Retain | It is critical to understand lymphatic spread, which affects prognosis and possibly the spread. |
| Survival Months | Retain | This is a secondary target variable, |
| Mortality Status | Retain | This is the target variable and needs to be kept. |

1. **Data Understanding**

**Task 2: Statistical Summary and Target Distribution**

*A screenshot of a table

Description automatically generated****Statistical Description Output:***

***Variable Scale Type:***

*A screenshot of a computer

Description automatically generated*

***Target Variable Distribution****:*

A graph with a red square

Description automatically generated

1. **Data Preparation**

**Task 3: Data Issues and Solutions**

***3a.***

***Issues Identification****: Summarize data quality issues discovered in the retained dataset.*

| Variable Name | Issue Description | Proposed Mitigation | Justification for Used Mitigation |
| --- | --- | --- | --- |
| 1. Age | Missing Values | Nine Total Missing Values | Nine missing values out of a total of four thousand and twenty-four instances equal 0.002%. Smaller than five percent. Delete missing instances |
| 2. Age | Float Values | Age needs to be integer not float | We need to ensure that age is not a float but instead an integer value. Convert columns to integers. |
| 3. Age | Max Value 502 | Age is unrealistic at five times the century. | One value out of four thousand and twenty-four instances equal 0.0002%. Smaller than five percent. Delete the instance |
| 4. Differentiate | Follow existing format | Change the column name to 'Differentiate' | Inconsistent data formatting seems unprofessional |
| 5. Mortality\_  Status | Follow existing format | Change the column name to 'Mortality Status' | Inconsistent data formatting seems unprofessional. Remove the underscore between the words |
| 6. Mortality\_  Status | Multiple output case strings | Alive vs ALIVE vs alive, Dead vs DEAD vs dead | Target output must have case consistency. Rename all to either 'Alive' or 'Dead' |
| 7. Mortality\_  Status | Missing Values | Four Total Missing Values | Four missing values out of a total of four thousand and twenty-four instances equal 0.09%. Smaller than five percent. Delete missing instances |
| 8. Regional Node Examined | Missing Values | One Total Missing Value | One value out of four thousand and twenty-four instances equal 0.0002%. Smaller than five percent. Delete the missing instance |
| 9. Regional Node Examined | Has a string value instead of int | Change 'One' instead of 1 | Ensuring that the data type is consistent with the output will improve model predictability. No indexing issues. |
| 10. Regional Node Examined | Has max value of 1080. 1080 Nodes Examined? | Remove instance. This value is more than 57 z-score points from the mean. Definitely an error. | One value out of a total of four thousand and twenty-four instances equals 0.0002%. Smaller than five percent. Delete the instance |
| 11. Reginol Node Positive | Space after Positive and fix spelling | Change to 'Regional Node Positive' | This could lead to incorrectly specifying the column. Notebook script issues. |
| 12. Regional Node Positive | Has a string value instead of int | Change 'One' instead of 1 | Ensuring that the data type is consistent with the output will improve model predictability. No indexing issues. |
| 13. Survival Months | Can't be more than age in months | 'Survival Months' calculated in years cannot be more than the age of the participant | One value out of four thousand and twenty-four instances equal 0.0002%. Smaller than five percent. Delete the instance |
| 14. Tumor Size | Missing Values | Three Total Missing Values | Three missing values out of a total of four thousand and twenty-four instances equal 0.0007%. Smaller than five percent. Delete missing instances |
| 15. Tumor Size | Tumor Size negative value | Tumor Size cannot have a negative value | One value out of four thousand and twenty-four instances equal 0.0002%. Smaller than five percent. Delete the instance |
| EXTRA | Removed total of 22 Instances | Percentage instances removed = 22/4024  0.01% | The total instances we removed was 0.01% which is good enough for the amount of data we have. We did not remove smaller outliers only unrealistic ones. |

**3b.**

**Resolution Implementation**: Provide brief explanations of the transformations applied and include

screenshots of key results if required.

****

1. **Before: After:**

**Solution:** *Remove 9 missing values. The size of the dataframe is big enough: 9/4024 = less than 5%. Check total removed after end of fixes*

**Code used:** *df.isnull().sum() # Sum up all the null values in data frame df*

*# Drop all instances that have a na column and reset index to ensure change*

*df = df.dropna().reset\_index(drop=True)*

1. **B: A:**

**Solution:** Change Age column from float to int

**Code used:** *df.loc[:, 'Age'] = df['Age'].astype(int) # Convert Age column to*

****

1. **B:** **A:**

**Solution:** *Remove single value*

**Code used:** *df.describe() # Describe function displays statistics*

*# Age row cannot have value 502, remove and reset index to ensure change*

*df = df[df['Age'] != 502]* *.reset\_index(drop=True)*



1. **B: A:**

**Solution:** *Change column name*

**Code used:** *# In data frame df rename column name.*

*df = df.rename(columns={'differentiate':"Differentiate"})*



1. **B: A:**

**Solution:** *Rename the column heading to the same format as other columns “\_”*

**Code used: *#*** *In data frame df rename column name.*

*df = df.rename(columns={'Mortality\_Status':"Mortality Status"})*

****

1. **B:**

****

**A:**

**Solution:** *Display all unique values of Mortality Status. Remove duplicate cases*

**Code used:** *# Make a copy and reset index before df.loc commands*

*df = df.copy().reset\_index(drop=True)*

*# Change all values to either Alive or Dead. df.loc is the function:*

*#df.loc[<row selection>, <column selection>] = <value>*

*df.loc[df['Mortality Status'] == 'ALIVE', 'Mortality Status'] = 'Alive'*

*df.loc[df['Mortality Status'] == 'alive', 'Mortality Status'] = 'Alive'*

*df.loc[df['Mortality Status'] == 'DEAD', 'Mortality Status'] = 'Dead'*

*df.loc[df['Mortality Status'] == 'dead', 'Mortality Status'] = 'Dead'*

1. **B:**  **A:**

**Solution:** *Remove 4 instances which include the missing values*

**Code used:** *df.isnull().sum() # Total the null values in each column*

*print(df.shape) # Print the amount of instances to compare after the drop*

*# Drop or delete all instances with null values and reset indext to ensure change.*

*df = df.dropna().reset\_index(Drop = True)*

*print(df.shape) #Print the amount of instances to compare after the drop*

***# 4024 – 4007 = 17 Values; 17 / 4027 = 0.4% (Less than 5%)***

1. **B: A:**

**Solution:** *Remove 9 missing values. The size of the dataframe is big enough: 9/4024 = less than 5%. Check total removed after end of fixes*

**Code used:** *df.isnull().sum() # Sum up all the null values in data frame df*

*# Drop all instances that have a na column and reset indext to ensure change.*

*df = df.copy().reset\_index(drop=True)*

****

1. **B: A:**

**Solution:** *Do a quick scan of data: Change number from ‘One’ to 1: From string to int*

**Code used:** *categorical\_columns = df.select\_dtypes(include='object').columns*

*# Create new variable and call all the columns with data types equal ‘object’*

*for column in categorical\_columns:*

*# Create a for loop for columns in the new variable categorical\_columns*

*display(df[column].value\_counts())*

*# For each column count all the values and summarize*

*print("\n")*

*# Change value from One to 1*

*df['Regional Note Positive'] = df['Regional Note Positive'].replace('One', 1)*

*# Convert column to int after changing One to 1*

*df['Regional Note Positive'] = df['Regional Note Positive'].astype(int)*

*#Verify Changes*

*print(df[['Regional Note Positive']].dtypes)*

*print(df[['Regional Note Positive']].value\_counts())*

****

1. **A:**

****

**B:**

**Solution:** *Check the stats again to find strange numbers. Noticed large max number compared to mean. Max = 1080 , Mean = 15. Node Examined cant be 1080!*

**Code used:** *std = 18.689170 # Value given df.describe()*

*value = 1080 # Value given df.describe()*

*mean = 14.62 # Value given df.describe()*

*z\_score = (value - mean) / std # Calculate Z-Score or points away from mean*

*z\_score = 57.01*

*# Remove instance with a value of 1080 in Regional Node Examined column and reset index to ensure change*

*df = df[df['Regional Node Examined'] != 1080].reset\_index(drop=True)*

*# Check statistics applied*

*df.describe()*



1. **B: A:**

**Solution:** Change column name

**Code used:** *# In data frame df rename column name*

*df = df.rename(columns={'Reginol Node Positive ':"Regional Node Positive"})*

1.  **B: A:**

**Solution:** *Do a quick scan of data: Change number from ‘One’ to 1: From string to int*

**Code used:** *categorical\_columns = df.select\_dtypes(include='object').columns*

*# Create new variable and call all the columns with data types equal ‘object’*

*for column in categorical\_columns:*

*# Create a for loop for columns in the new variable categorical\_columns*

*display(df[column].value\_counts())*

*# For each column count all the values and summarize*

*print("\n")*

*# Change value from One to 1*

*df['Regional Note Positive'] = df['Regional Note Positive'].replace('One', 1)*

*# Convert column to int after changing One to 1*

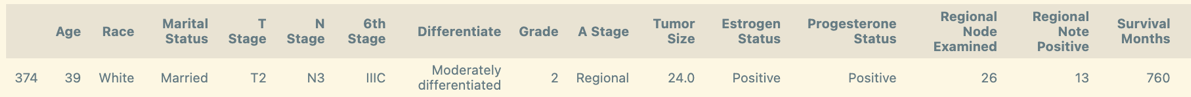
*df['Regional Note Positive'] = df['Regional Note Positive'].astype(int)*

*#Verify Changes*

*print(df[['Regional Note Positive']].dtypes)*

*print(df[['Regional Note Positive']].value\_counts())*

1. ** B:**

****

****

**A:**

**Solution:** *Check the stats and find strange numbers. Noticed 760 months, which is 63.33 years. Confirm participant’s age.*

**Code used:** *df.describe() # Display statistics in data frame df.*

*# Create outlier\_row variable and display the single line that has a Survival Month value that equals 760. Reset index to ensure change in df.*

*outlier\_row = df[df['Survival Months'] == 760]*

*outlier\_row # Display the outlier row*

*# Remove instance in df where Survival Month value equals 760 and reset index to ensure change in df.*

*df = df[df['Survival Months'] != 760] .reset\_index(drop=True)*

*df.describe() # Check if this value has been removed*

****

1. ** B: A:**

**Solution:** *Remove 3 missing values. The size of the dataframe is big enough: 3/4024 = less than 5%. Check total removed after end of fixes*

**Code used:** *df.isnull().sum() # Sum up all the null values in data frame df*

*df = df.dropna() # Drop all instances that have a na column*

1. ** B:**

****

**A:**

**Solution:** *Check the stats and find strange numbers. Noticed a negative number for tumor size. This is not possible. Remove instance that contains negative tumor size.*

**Code used:** *df.describe() # Display statistics in data frame df.*

*# Change dataframe’s tumor size column to not include -75 and reset index to ensure change*

*df = df[df['Tumor Size'] != -75] .reset\_index(drop=True)*

*df.describe() # Check if this value has been removed*

1. **Modelling**

**Task 4: Classification Algorithms and Model Construction**

**4a.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algorithm Name | Algorithm Type | Learnable Parameters | Some Possible Hyperparameters | Imported Python package to use the algorithm |
| NB | Parametric | Mean, Variance, Class Probabilities | Distribution type: Gaussian Naïve Bayes, Multinomial Naïve Bayes, Bernoilli Naïve Bayes | *sklearn.naive\_bayes import GausianNB, MultinomialNB* |
| LR | Parametric | Weights or co-efficient, Bias (intercept term) | Penalty, to prevent overfitting and Solver which is an algorithm to optimisation. | *sklearn.linear\_model import LogisticRegression* |
| KNN | Non-Parametric | No learnable parameters, its non-parametric | Number of Neighbors, Distance Metric (Euclidean, Manhattan), Weights (Uniform or distance-based), Value of K, odd numbers, 1, 3, 5 | *sklearn.neighbors import KNeighborsClassifier* |

**4b.**

**Model Construction**:

**A screenshot of a computer

Description automatically generated**

1. **A black background with a black square

   Description automatically generated with medium confidence**We have 4003 instances within our dataset. According to Joseph (2022), the optimal training-test split is 88.8% training data and 11.2% testing data. The optimal training-test split ratio is determined by **** , where *p* is the number of parameters in the dataset. If *p* is unknown, we can use *Nu*, which indicates the number of unique instances in the dataset. Therefore:

****

**; ;**

1. The training-test approach to modelling, effectively splits data for probability calculations. We use a training dataset with labelled outputs to create learnable values, then test the model with a similar dataset to evaluate its performance without labels. After the test we compare the output of the labels with the actual labels. K-fold cross-validation is another method that divides the dataset into multiple training and test sets, optimizing learning potential and reducing overfitting. While standard training-test splits are used for probability calculations, cross-validation is more commonly used for testing and tuning hyperparameters in GridSearch.
2. **Training and Test Split Code:**

*# Define the variables, specify the test\_size in percentage, define random\_state for reproducability and Stratify the data to ensure the output variables are equally split.*

*X\_train\_knn\_maxmin, X\_test\_knn\_maxmin, y\_train\_knn\_maxmin, y\_test\_knn\_maxmin = train\_test\_split(X\_maxmin\_knn, y,test\_size = 0.25, random\_state = 42, stratify = y)*

*This is an example for splitting the training and testing data for KNN with the data frame where maxmin scaling was used to impute the data frame.The split here is defined by the test\_size of 0.25:*

*Training = 75%*

*Test = 25%*

1. **Stratifying Data:**

*stratify = y*

*This ensures that if we have 100 instances with classification output: Dead / Alive as the following:*

**If you do a 70-30 train/test split with stratification:**

**Training Set (70 patients):** # 70%

56 Alive (80% of 70)

14 Dead (20% of 70)

**Test Set (30 patients):** # 30%

24 Alive (80% of 30)

6 Dead (20% of 30)

1. **Model Performance Analysis**

**Task 5: Evaluation: How good are your models**

**5a.**

**Out of the 6 models, two for each algorithm (One scaled at MaxMin for continuous numeric values and the other Adjusted based on distribution), these are the three that performed the best based on the following criteria:**

**True Positive:** Focus on mainly high recall but also precision. We need to ensure that we don’t have too many True Negatives. We don’t want to tell too many people that they are sick and need treatment when in fact they don’t.

Precision = TP / (TP + FP)

**False Negative:** Focus on high Recall value. Use when False Negatives are costly. In this case we don’t want to miss patients that are ill.

Recall = TP / (TP + FN)

**1. KNN | Adjusted Scaling**

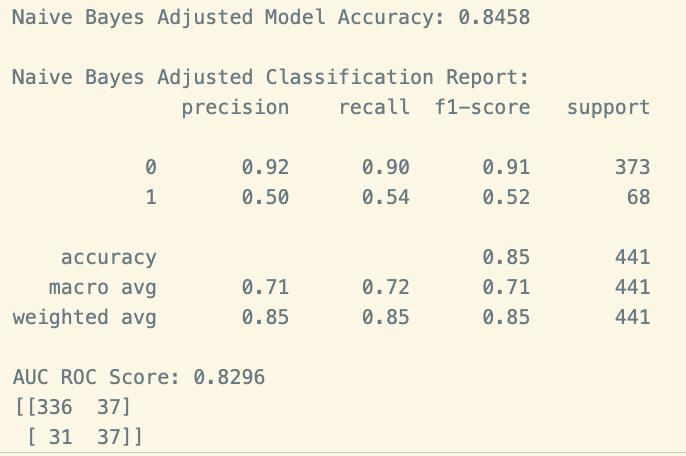
**A blue and white diagram

Description automatically generatedA screenshot of a computer

Description automatically generated**

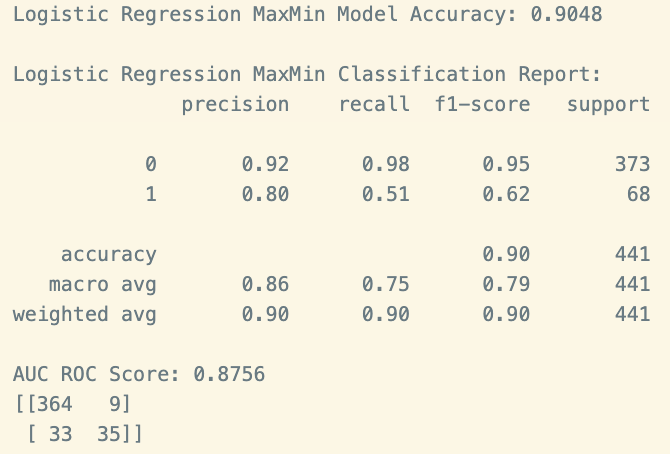
***5a. (continue)***

**2. Naïve Bayes | Adjusted Scaling**

**A diagram of a confused matrix

Description automatically generated**

**3. Logical Regression | MinMax Scaling**

**A blue and white graph

Description automatically generated**

**5b.**

| Metrics | USE or DO NOT USE | Justification in relation to the success criteria | Model Name | Test Score |
| --- | --- | --- | --- | --- |
| Accuracy | DO NOT USE | The accuracy is not important. Focus on False Negatives and True Negatives | NB | 0.85 |
| DO NOT USE | The accuracy is not important. Focus on False Negatives and True Negatives | LR | 0.9 |
| DO NOT USE | The accuracy is not important. Focus on False Negatives and True Negatives | KNN (K=5) | 0.88 |
| Recall | USE | Recall is the most important, False Negatives focus. Above 0.5 for [1] | NB | [0] 0.90, [1] 0.54 |
| USE | Recall is the most important, False Negatives focus. Above 0.5 for [1] | LR | [0] 0.98, [1] 0.51 |
| DO NOT USE | Recall is the most important, False Negatives focus. Below 0.5 for [1] | KNN (K=5) | [0] 0.98, [1] 0.34 |
| Precision | DO NOT USE | Precision is important for balance, False Positive Focus. At 0.5 for [1] | NB | [0] 0.92, [1] 0.50 |
| USE | Precision is important for balance, False Positive Focus. Above 0.5 for [1] | LR | [0] 0.92, [1] 0.80 |
| USE | Precision is important for balance, False Positive Focus. Above 0.5 for [1] | KNN (K=5) | [0] 0.89, [1] 0.74 |
| F-Score | USE | F-Score is harmony between Recall and Precision. Above 0.5 for [1] | NB | [0] 0.91, [1] 0.52 |
| USE | F-Score is harmony between Recall and Precision. Above 0.5 for [1] | LR | [0] 0.95, [1] 0.62 |
| DO NOT USE | F-Score is harmony between Recall and Precision. Below 0.5 for [1] | KNN (K=5) | [0] 0.93, [1] 0.46 |
| AUC-ROC | USE | Excellent Discrimination across all classes and their thresholds | NB | 0.83 |
| USE | Excellent Discrimination across all classes and their thresholds | LR | 0.88 |
| DO NOT USE | Acceptable Discrimination across all classes and their thresholds | KNN (K=5) | 0.79 |

**5c.**

Out of the three classification models, logistical regression featured in 5 'USE' cases, while Naïve Bayes featured in 4. The logistic regression had the highest overall scores in most performance metrics. We needed to focus on the 'dead class' to ensure we are not sending people away as a false negative (we need a low false negative score as this is costly); instead, we should check on patients overcompensating for false positives more often. Based on the performance metrics evaluated in Task 5b, the Logistic Regression model with MinMax Scaling is the single best classification model for this scenario. It achieved the highest accuracy (0.9048), precision (0.80 for "Dead"), and recall (0.51 for "Dead"), alongside an AUC ROC score of 0.8756.

**5d.**

**A graph of a graph

Description automatically generated with medium confidence**

The Precision-Recall Curve gives us a good idea of how the model is performing as the clinicians have indicated a low false negative score but balancing false positives. The model performs well on precision when the recall is low, but as we increase the recall the precision drops off very quickly. The chart is jiggered, indicating the instability of the model’s predictions. As the recall number is only 51% in this model, we would be hesitant to say that this is a good fit model and ready for a clinic setting. The model is underfit for its purpose.

**5e.**

1. The number of folds we use was **10.**
2. **Fitting 10 folds for each of 10 candidates, totalling 100 fits**

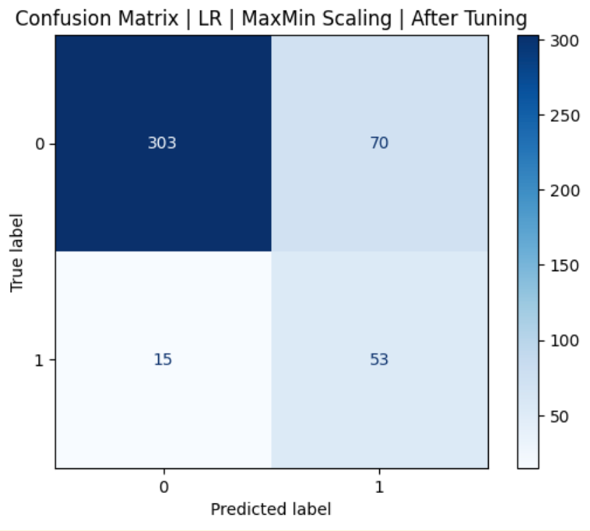
Logistic Regression MaxMin Best Parameters: {'C': 10, ’class weight’: ‘balanced’, 'penalty': 'l2', 'solver': 'lbfgs'}

* **C:** *Penalize overfitting: high number = overfitting risk, low number = underfitting risk*
* **Class Weight:** Adjusts weights within class inversely to the frequency of the class.
* **Penalty:** *Add penalty to co-efficients, preventing them from becoming too large and overfit. L2 is Default in SciKit Learn*
* ***Solver:*** *Had to research, works with L2*

Logistic Regression MaxMin Best Cross-Validation Accuracy: 75.79%

1. **Before Tuning After Tuning**

A blue and white graph

Description automatically generated

We sacrificed a reduction in **false negatives** for **false positives**, which is an improved fit to the criteria of the clinicians based on the criteria.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Metrics | USE or DO NOT USE | Justification in relation to the success criteria | Model Name | Test Score |
| Accuracy | DO NOT USE | The accuracy is not important. Focus on False Negatives and True Negatives | LR | 0.81 |
| Recall | USE | Recall is the most important, False Negatives focus. Above 0.5 for [1] | LR | [0] 0.81, [1] 0.78 |
| Precision | USE | Precision is important for balance, False Positive Focus. Above 0.5 for [1] | LR | [0] 0.95, [1] 0.43 |
| F-Score | USE | F-Score is harmony between Recall and Precision. Above 0.5 for [1] | LR | [0] 0.88, [1] 0.55 |
| AUC ROC | USE | Exellent Discrimination across all classes and their thresholds | LR | 0.88 |

1. The hyper-tuning improved the best fit for the model. We wanted to reduce the recall value to ensure that we do not miss any breast cancer patients who might be sicker than they are. By doing that, we had to sacrifice precision, which was perfectly fine in that we, unfortunately, had to incorrectly let some cancer patients know that they were sicker than they were. It was a trade-off that was worth it in the end for the clinicians looking at the overall well-being of the patient, specifically the most vulnerable.

**5f.**

The logistic regression model attained an recall of 78% and an AUC-ROC of 0.8756 in mortality prediction which is the fitted criteria based on the clinicians requests. While hyperparameter tuning improved recall from 51% to 78%, accuracy is not the main concern here.

Key limitations include:

1. A small dataset of 4,003 cases.

2. Potential demographic biases regarding race.

3. A precision-recall trade-off.

Logistic regression outperformed KNN and Naive Bayes due to its effectiveness with categorical features. Ethically, false positives may cause psychological distress and unnecessary treatments, while false negatives could delay critical care. Thus, this model should support, not replace, clinical care.

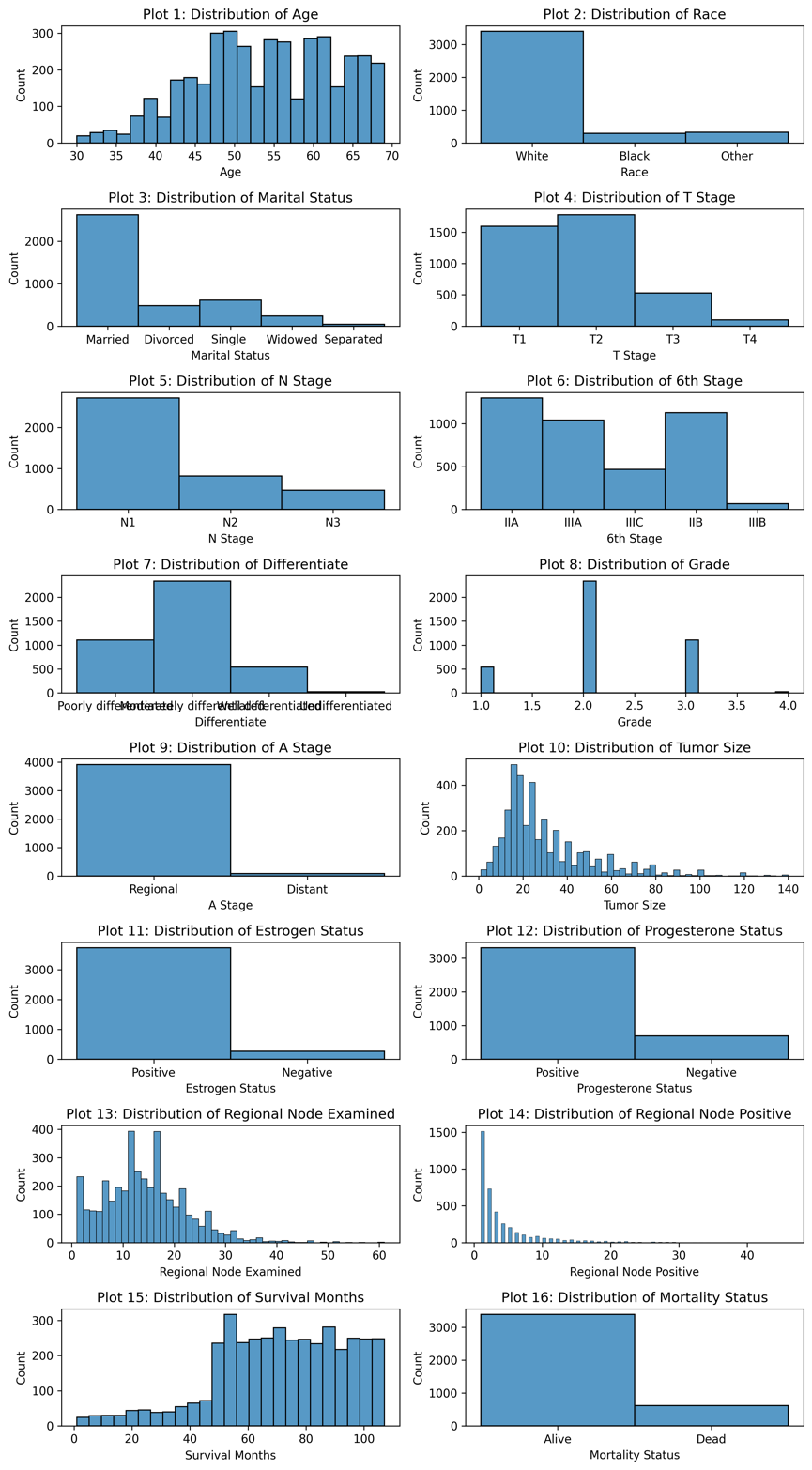
**PART B: Breast Cancer Survival Rate Prediction**

**Task 1: Domain Understanding: Regression**

**A screen shot of a computer

Description automatically generated**

**Task 2: Data Understanding: Producing Your Experimental Designing**

****

**Task 3: Data Preprocessing: Transforming your data**

**3a.**

There are 5 feature I would like to propose scaling, and they include the following along with the motivations behind the scaling:

|  |  |  |
| --- | --- | --- |
| Feature | Type of Scaling | Motivation behind Scaling |
| Age | Standard Scaling | The Age feature is well distributed in a close to normal way |
| Tumor Size | Logarithmic Scaling | Tumor size has many outliers and is skewed to the right |
| Regional Node Examined | Logarithmic Scaling | Regional Node Examined has many outliers and is skewed to the right |
| Regional Node Positive | Logarithmic Scaling | Regional Node Positive has many outliers and is skewed to the right |
| Survival Months | Robusta Scaling | Survival Months have less skewing, however not very normally distributed |

**Scaled\_df.describe()**

If we take a look at the mean and the standard deviation, we can see that across the five features they differ quite enough for us to ensure scaling and normalization

**A screenshot of a computer

Description automatically generated**

**3b.**

We would only scale continuous numeric values.

**Task 4: Modelling: Build Predictive Regression Models**

**4a.**

Decision Trees are highly interpretable algorithms that make them exceptionally usable in a healthcare setting where outcome-based explanations or experiments must be simple to explain and replicate. Decision trees can utilize a combination of numeric and categorical data within a dataset and are very easy to compute. (47 words)

**4b(i).**

model2\_update = DecisionTreeRegressor(random\_state = 10, max\_depth = 4)

**random\_state = 10** # This code is what makes the model reproduceable

**4b(ii).**

*# Dimensions for Model 1 training and test*

print(**f**"Model 1 Training-Set Dimensions: {X\_train\_model1.shape}, {y\_train\_model1.shape}")

print(**f**"Model 1 Test-Set Dimensions: {X\_test\_model1.shape}, {y\_test\_model1.shape}")

*# Display Model 1 features*

features\_m1 = ['Age', 'Tumor Size', 'Regional Node Examined', 'Regional Node Positive']

print(**f**"Features used in Model 1: {features\_m1}")

*# Dimensions for Model 2 training and stst*

print(**f**"Model 2 Training-et Dimensions: {X\_train\_model2.shape}, {y\_train\_model2.shape}")

print(**f**"Model 2 Test-Set Dimensions: {X\_test\_model2.shape}, {y\_test\_model2.shape}")

*# Display Model 2 features*

features\_m2 = list(encoded\_df.columns.drop(target))

print(**f**"Features used in Model 2: {features\_m2}")

**Output:**

**Model 1 Training-Set Dimensions:** (3202, 4), (3202,)

**Model 1 Test-Set Dimensions:** (801, 4), (801,)

**Features used in Model 1:** ['Age', 'Tumor Size', 'Regional Node Examined', 'Regional Node Positive']

**Model 2 Training-et Dimensions**: (3202, 15), (3202,)

**Model 2 Test-Set Dimensions:** (801, 15), (801,)

**Features used in Model 2:** ['Age', 'Race', 'Marital Status', 'T Stage', 'N Stage', '6th Stage', 'Differentiate', 'Grade', 'A Stage', 'Tumor Size', 'Estrogen Status', 'Progesterone Status', 'Regional Node Examined', 'Regional Node Positive', 'Mortality Status']

**Task 5: Evaluation: How good are your Models**

**5a.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Metrics | USE or  DO NOT USE | Justification in relation to the  success criteria | Model Name | Test Score |
| MSE | DO NOT USE | The value is incredibly high and is difficult to evaluate | Model 1 | 1,056.78 |
| DO NOT USE | The value is slightly lower but still high, and remains difficult to interoperate | Model 2 | 750.64 |
| MAE | DO NOT USE | Possibly the best and most interoperable error value | Model 1 | 25.83 |
| DO NOT USE | Possibly the best and most interoperable error value | Model 2 | 22.21 |
| R-Squared | DO NOT USE | A Negative R-Squared is incredibly weak, close to -1 | Model 1 | (1.06) |
| DO NOT USE | A Negative R-Squared is not good at all | Model 2 | (0.46) |

**5b.**

Both models had a negative R-squared, indicating that the model fits more noise than the signal. The MAE, the most accessible metric to interpret, states a possible error of almost 22 months, approximately two years. This interpretation of predictability could be much better.

**5c.**

The single best model is Model 2. The MAE of Model 2 is lower, along with the MSE and the R-squared being closer to 1. However, we need to be more comfortable utilising these models in real-world clinical studies. The negative R-squared value indicates a low predictive relationship or minimal connection between the features and the ability to predict Survival Months. The MSE is incredibly high and only relevant to clinic understanding by square rooting the value and instead using RMSE. The only value relevant in a clinical setting is the MAE. One out of three statistical metrics does not constitute as usable.

**A diagram of a company

Description automatically generated5d.**

**5e.**

**76 Months = 6 years and 4 months**

**References**

*(List all references alphabetically in Harvard style)*

Example:

* Joseph, V.R., 2022. Optimal ratio for data splitting. *Statistical Analysis and Data Mining: The ASA Data Science Journal,* 15(4), pp.531–538. Available at: https://doi.org/10.1002/sam.11583 [Accessed 21 November 2024].