PREDICTIVE ANALYSIS OF BREAST CANCER SURVIVAL STATUS BASED ON THE TUMOR ATTRIBUTES

Presentation by
Group – 06

Haneesha Chowdary Donepudi,
Kalyan Raj Chinigi,
Kavya Surabhi,
Priyankitha Kandi,
Saikumar Reddy Yalagalapalli
Srihari Myla Venkata,

INTRODUCTION

- Breast cancer is one of the most common cancers affecting women worldwide accounting for over 2 million new cases annually. It is a complex, heterogeneous disease with multiple risk factors contributing to its initiation and progression.

- The dataset contains information about the patient's medical history such as Tumor size, Lymph node status, Tumor grade, and survival status. Among these factors, hormonal imbalances play a significant role in breast cancer pathogenesis.

- The knowledge gathered from this study may be utilized to inform decisions about advanced breast cancer prevention, diagnosis, and therapy, as well as to forecast how well hormone therapies will work.

Aim

- " The primary objective of this study is to conduct predictive analysis correlating tumor attributes, including hormonal imbalances such as estrogen and progesterone levels, with the survival status of individuals diagnosed with breast cancer".

Purpose

- The purpose of this study is to examine a dataset of breast cancer patients to find the parameters linked with breast cancer survival rates. The dataset includes details regarding the patient's medical history, such as tumor size, hormone imbalances, tumor grade, and survival status. The study's goal is to create a predictive algorithm that can accurately forecast breast cancer survival rates based on medical data from patients. This study's findings could be used to create more effective treatment strategies and enhance patient outcomes.

RESEARCH HYPOTHESIS

1. **Null Hypothesis:**
   There is no significant correlation between the selected biological and clinical features, including hormonal imbalances, and the survival rates of individuals diagnosed with breast cancer.

2. **Alternative Hypothesis:**
   There exists a significant correlation between the selected biological and clinical features, including hormonal imbalances, and the survival rates of individuals diagnosed with breast cancer.

METHODOLOGY

1. **Data Collection:**
   - Breast Cancer.
   - The dataset was taken from Kaggle.
   - Link to Dataset https://www.kaggle.com/datasets/alaahussien/breast-cancer
2. **Data Storage:** Data stored in CSV file.
3. **Data Cleaning:** Performed necessary operations for handling null values.
4. **Changing Categorical to Numerical:** Categorical variables like ‘T Stage , 6th Stage, N Stage, Race, differentiate, Marital Status, Grade, A Stage, Estrogen Status, Progesterone Status and Status are converted into numerical ones for the machine to understand.
5. **Data Analysis and Visualization:** Visualizing the data.
6. **Hypothesis Testing:** Verifying the features.
7. **Model Development and Analysis:** Working on different ML models.
8. **Evaluating Results:** Deciding which model is performing the best.

Data cleaning
&
Encoding

- There are 16 attributes that are present in our CSV file, in our data set we haven't found any null values.

- Categorical variables like ‘T Stage , 6th Stage, N Stage, Race, differentiate, Marital Status, Grade, A Stage, Estrogen Status, Progesterone Status and Status are converted into numerical ones for the machine to understand by using Encoding.

```
df.dtypes
```

```
Age                        int64
Race                      object
Marital Status            object
T Stage                   object
N Stage                   object
6th Stage                 object
differentiate             object
Grade                     object
A Stage                   object
Tumor Size                 int64
Estrogen Status           object
Progesterone Status       object
Regional Node Examined     int64
Reginol Node Positive      int64
Survival Months            int64
Status                    object
dtype: object
```

```
# checking for null values
df.isnull().sum()

# we don't have any null values . Therefore Data Cleaning is not necessary
```

```
Age                       0
Race                      0
Marital Status            0
T Stage                   0
N Stage                   0
6th Stage                 0
differentiate             0
Grade                     0
A Stage                   0
Tumor Size                0
Estrogen Status           0
Progesterone Status       0
Regional Node Examined    0
Reginol Node Positive     0
Survival Months           0
Status                    0
dtype: int64
```

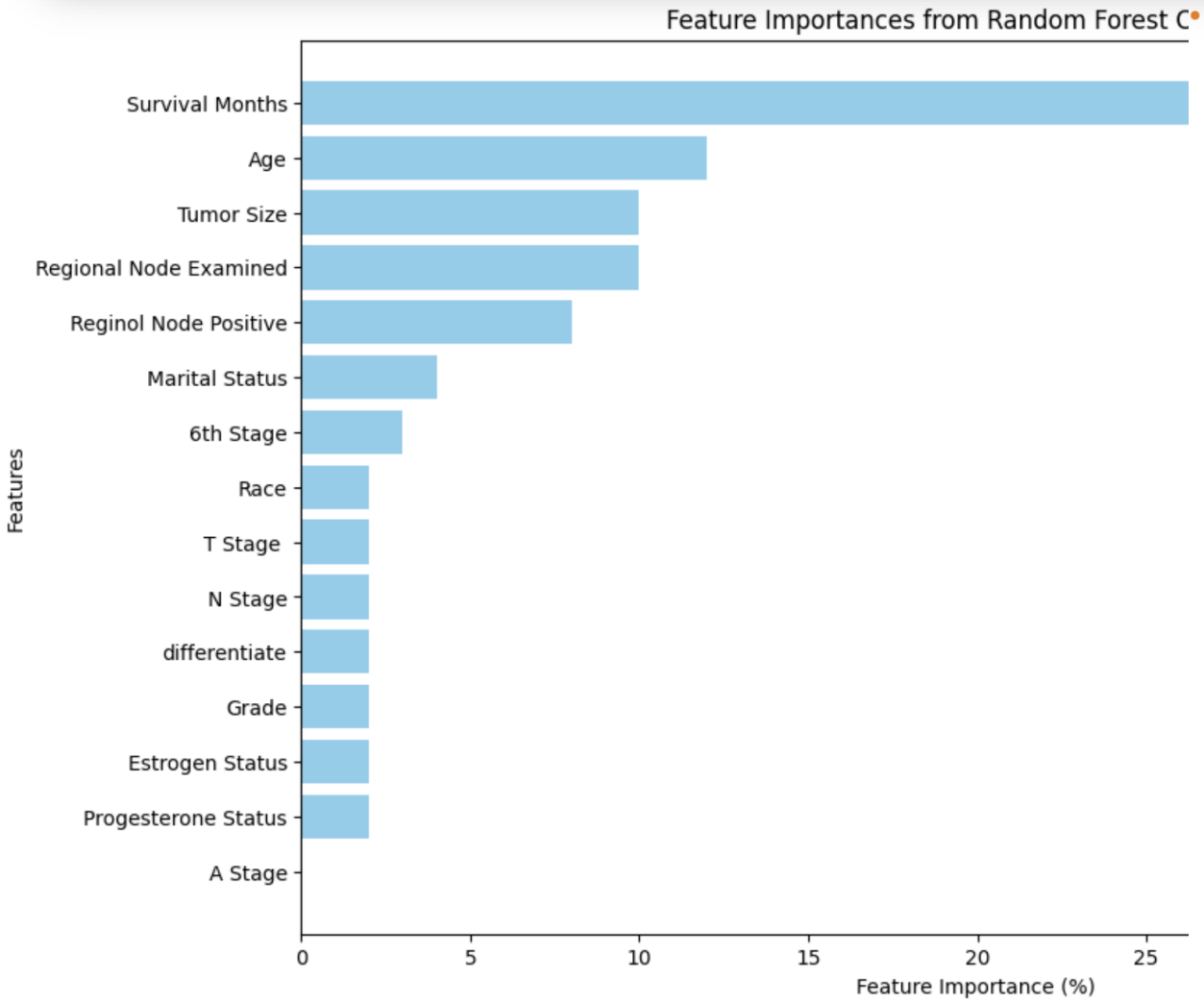
Changed to numerical

	Age	Race	Marital Status	T Stage	N Stage	6th Stage	differentiate	\
0	68	2	1	0	0	0	1	
1	50	2	1	1	1	2	0	
2	58	2	0	2	2	4	0	
3	58	2	1	0	0	0	1	
4	47	2	1	1	0	1	1	

	Grade	A Stage	Tumor Size	Estrogen Status	Progesterone Status	\
0	3	1	4	1	1	
1	2	1	35	1	1	
2	2	1	63	1	1	
3	3	1	18	1	1	
4	3	1	41	1	1	

	Regional Node Examined	Reginol Node Positive	Survival Months	Status
0	24	1	60	0
1	14	5	62	0
2	14	7	75	0
3	2	1	84	0
4	3	1	50	0

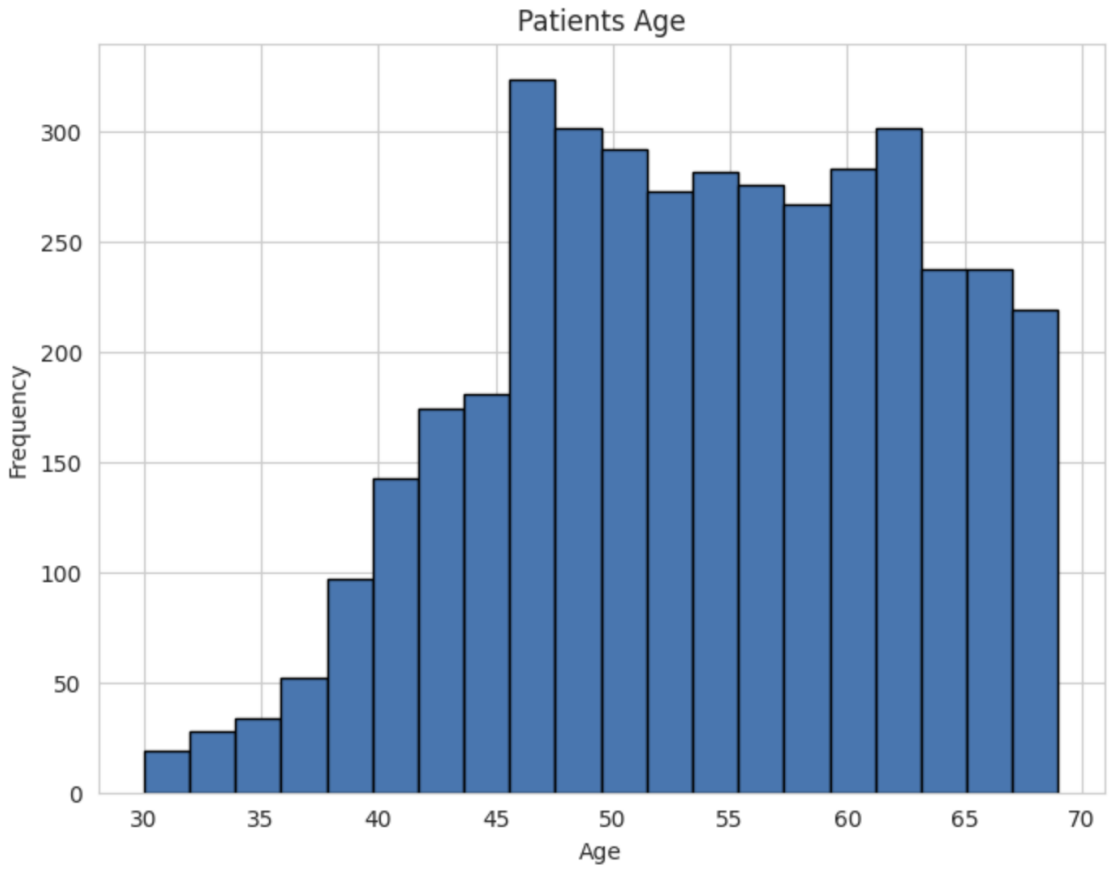
FEATURE IMPORTANCE

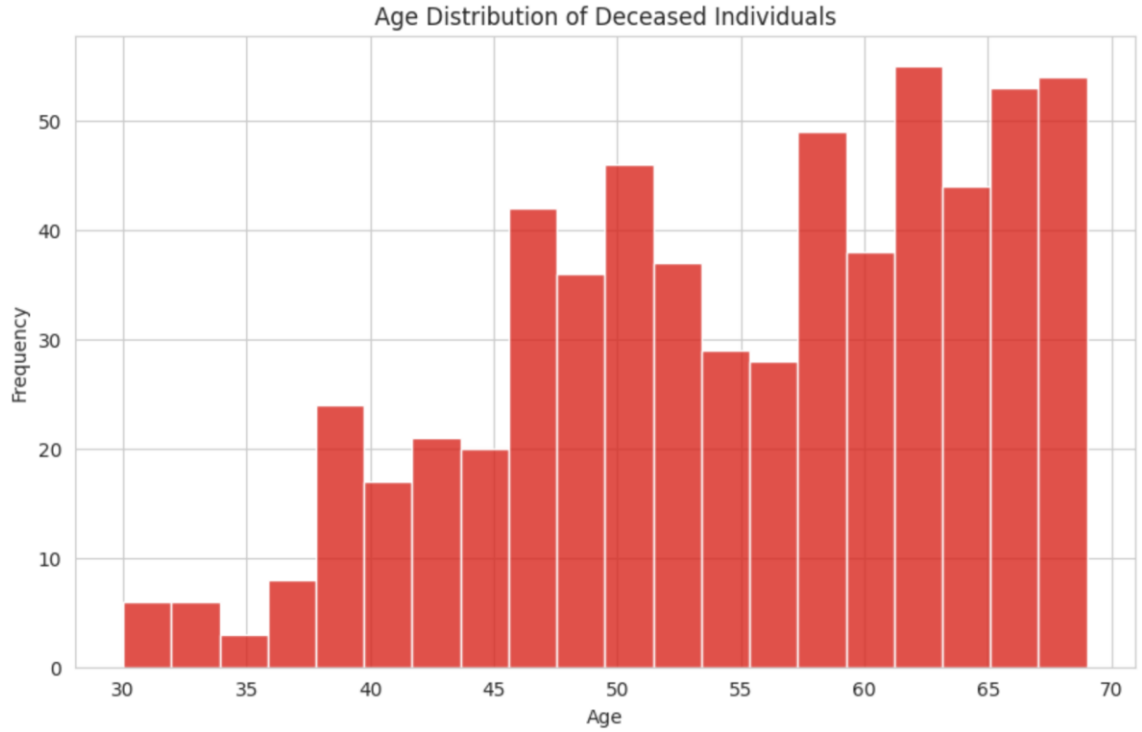


[21]:

```
### FEATURE IMPORTANCE
###  FEATURE IMPORTANCE DERIVED FROM RANDOM FOREST CLASSIFIER

RF_fit = RandomForestClassifier()
RF_fit.fit(X_train,y_train)

Coef_ = pd.DataFrame(X_train.columns, RF_fit.feature_importance
Coef_.columns  = ['Coef', 'Cols']
Coef_['Coef'] = round(Coef_.Coef*100, )
Coef_ = Coef_[['Cols', 'Coef']].sort_values('Coef', ascending =
Coef_['CumSum'] = round(Coef_.Coef.cumsum())
Coef_ = Coef_.reset_index().drop('index', axis = 1)
display(Coef_)

# Plotting
plt.figure(figsize=(12, 8))
plt.barh(Coef_['Cols'], Coef_['Coef'], color='skyblue')
plt.xlabel('Feature Importance (%)')
plt.ylabel('Features')
plt.title('Feature Importances from Random Forest Classifier')
plt.gca().invert_yaxis()  # To display the highest importance a
plt.show()
```

DATA VISUALIZATION



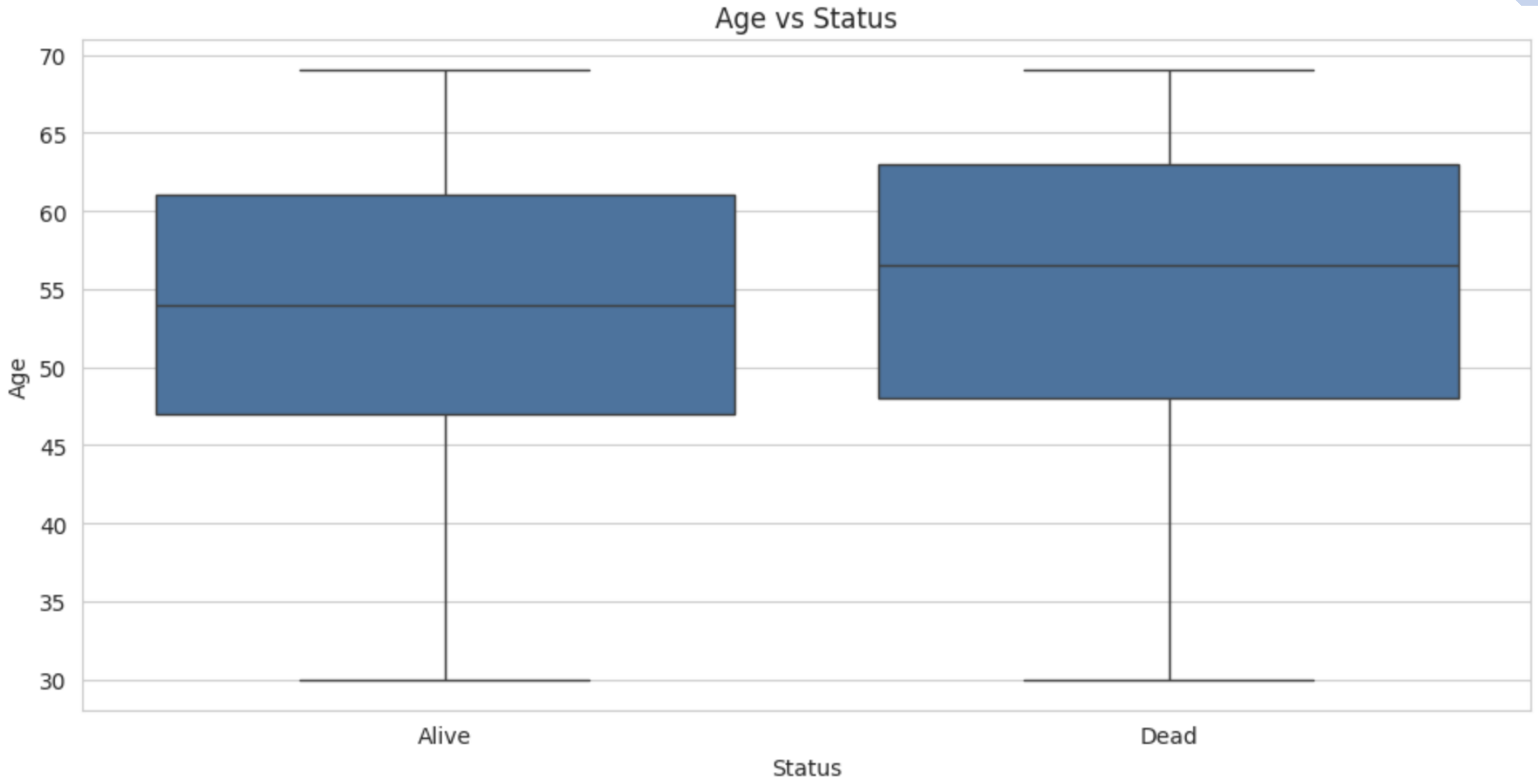


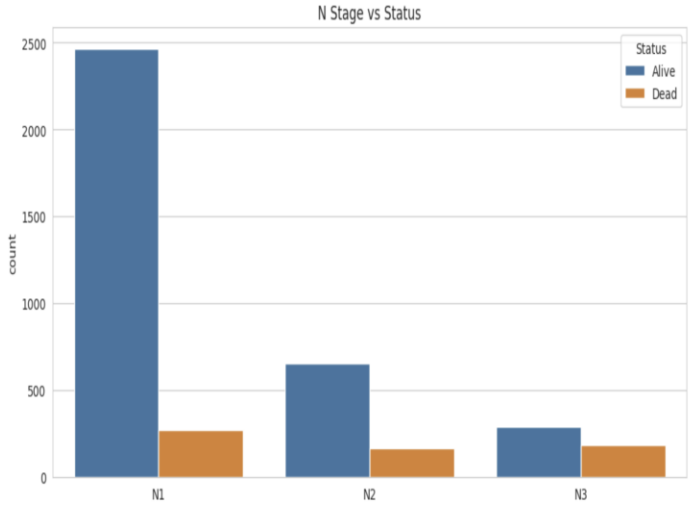
Representing Relationship b/w different features in the data frame and target column

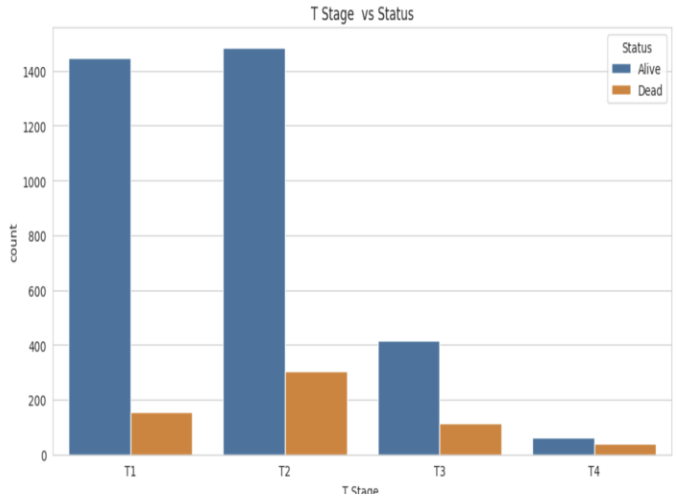
```
# 'Status' is target column
target_col = 'Status'
```

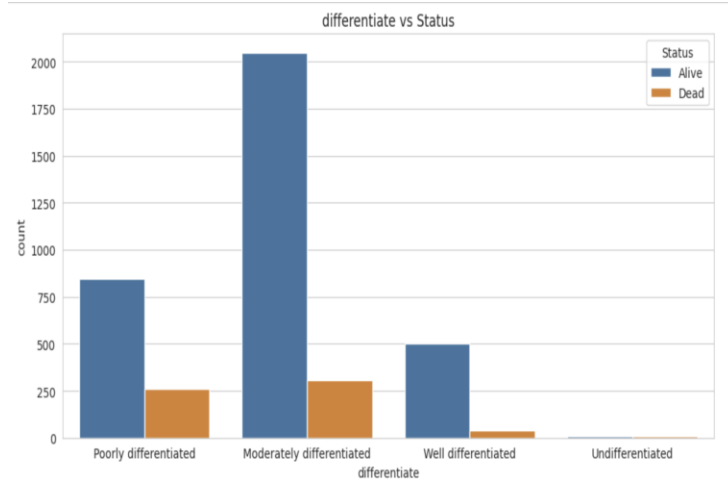
```
# Define the number of rows needed for subplots
num_rows = len(df.columns) - 1  # excluding the target column

# Create subplots
fig, axes = plt.subplots(nrows=num_rows, ncols=1, figsize=(10, 5 * num_rows))
```

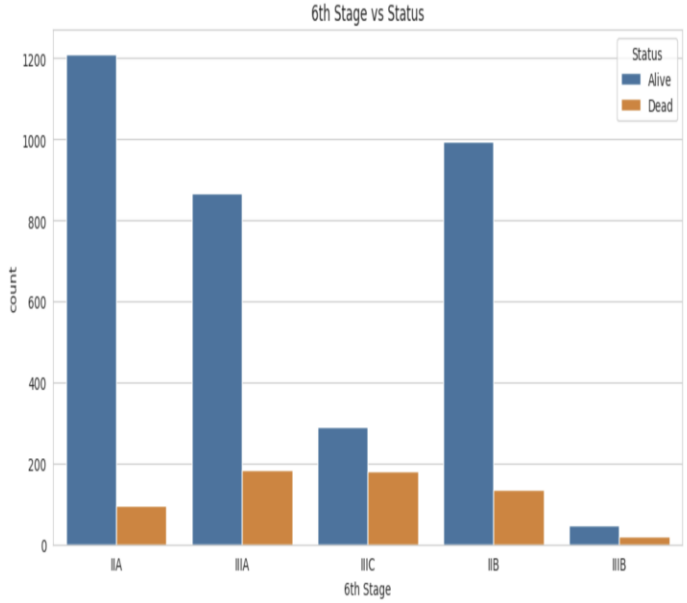
```
# Iterate through columns and create plots
for i, column in enumerate(df.columns):
    if column == target_col:
        continue  # Skip the target column

    ax = axes[i] if num_rows > 1 else axes  # Handle case of single subplot

    if column in numeric_cols:
        # For numeric columns, use a boxplot or scatterplot
        sns.boxplot(x=target_col, y=column, data=df, ax=ax)


    elif column in categorical_cols:
        # For categorical columns, use a countplot or barplot
        sns.countplot(x=column, hue=target_col, data=df, ax=ax)
        # Alternatively, for barplot: sns.barplot(x=target_col, y=column, data=df, ax=ax)

    ax.set_title(f'{column} vs {target_col}')
```

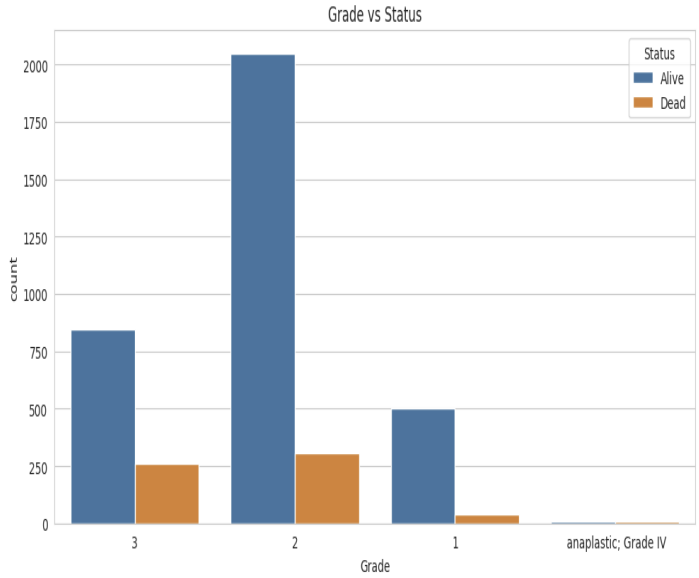

Age vs Status
70
65
60
55
50
45
40
35
30
Age
Alive
Dead
Status

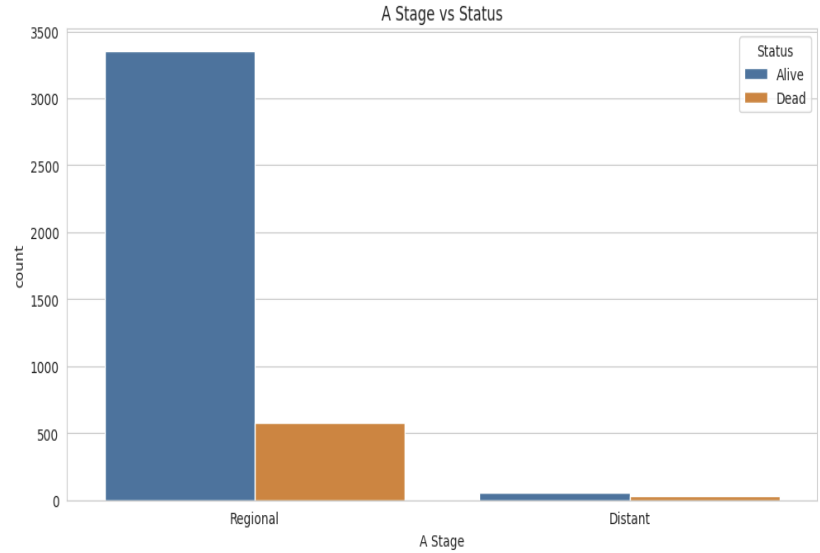

N Stage vs Status
Status
Alive
Dead
count
2500
2000
1500
1000
500
0
N1
N2
N3

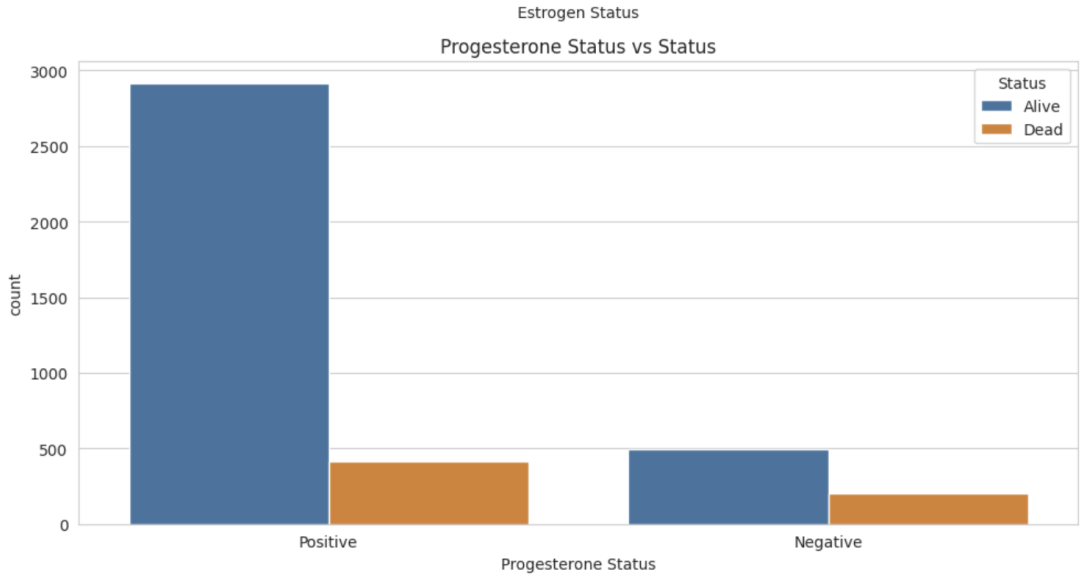

T Stage vs Status
Status
Alive
Dead
count
1400
1200
1000
800
600
400
200
0
T1
T2
T3
T4
T Stage

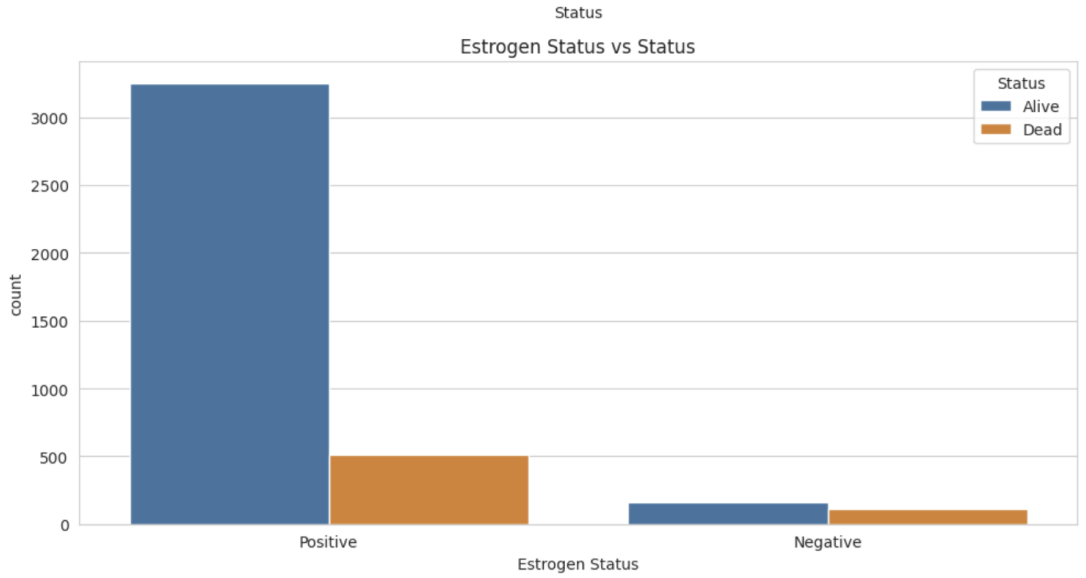

differentiate vs Status
Status
Alive
Dead
count
2000
1750
1500
1250
1000
750
500
250
0
Poorly differentiated
Moderately differentiated
Well differentiated
Undifferentiated
differentiate


6th Stage vs Status
Status
Alive
Dead
count
1200
1000
800
600
400
200
0
IIA
IIIA
IIIC
IIB
IIIB
6th Stage

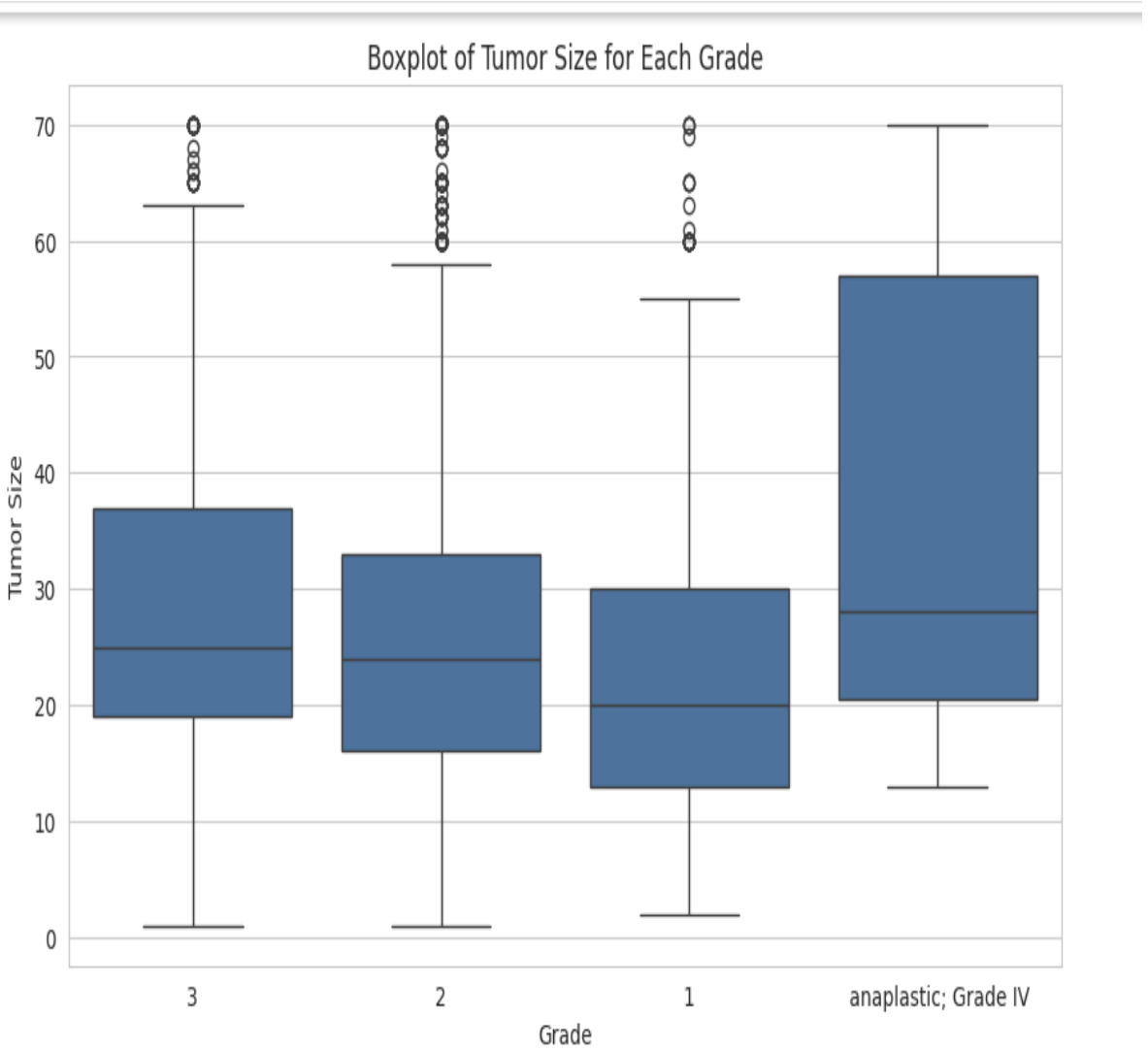

Grade vs Status
Status
Alive
Dead
count
2000
1750
1500
1250
1000
750
500
250
0
3
2
1
anaplastic; Grade IV
Grade


A Stage vs Status
Status
Alive
Dead
count
3500
3000
2500
2000
1500
1000
500
0
Regional
Distant
A Stage

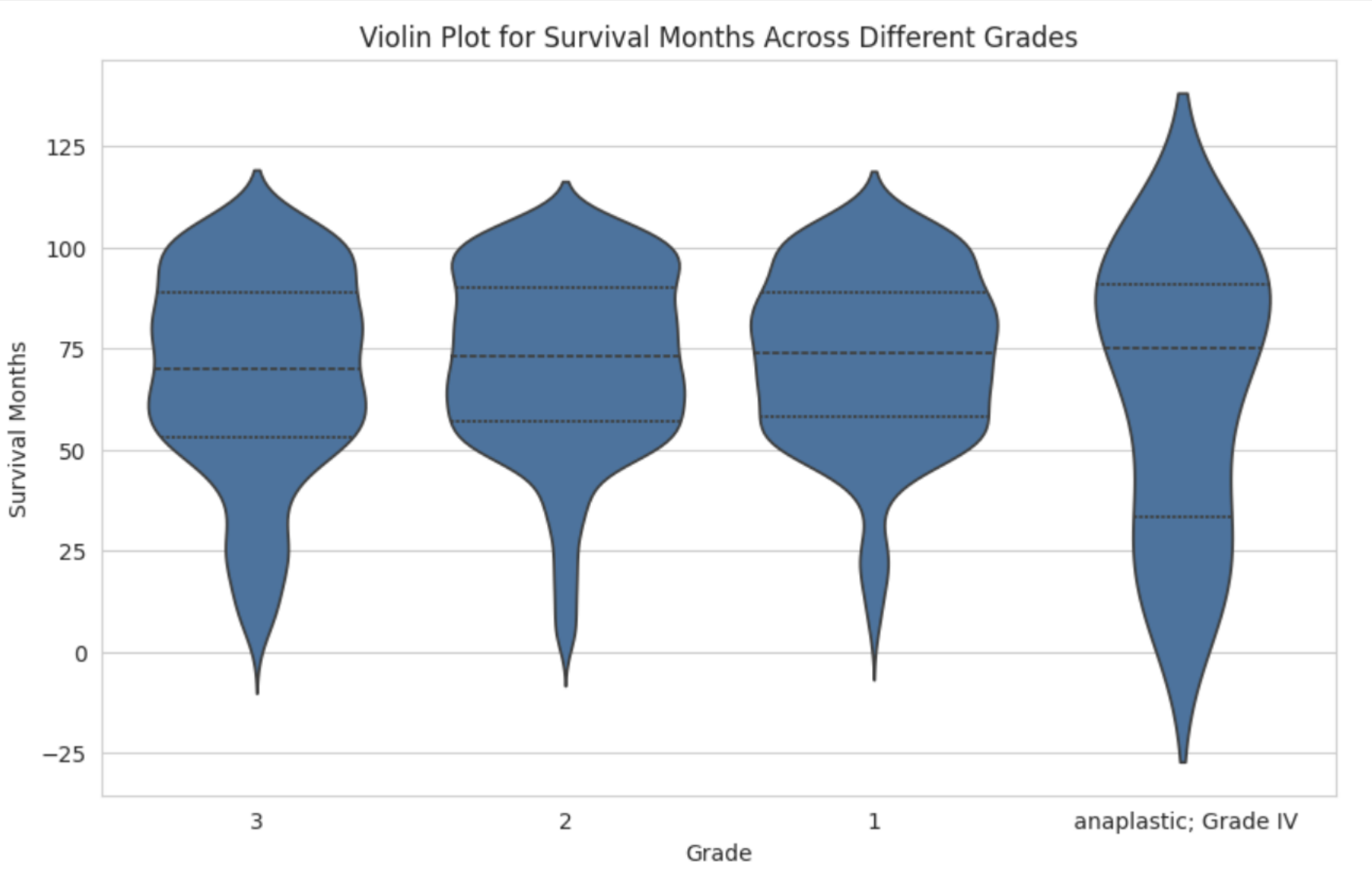

Estrogen Status
Progesterone Status vs Status
Status
Alive
Dead
count
3000
2500
2000
1500
1000
500
0
Positive
Negative
Progesterone Status

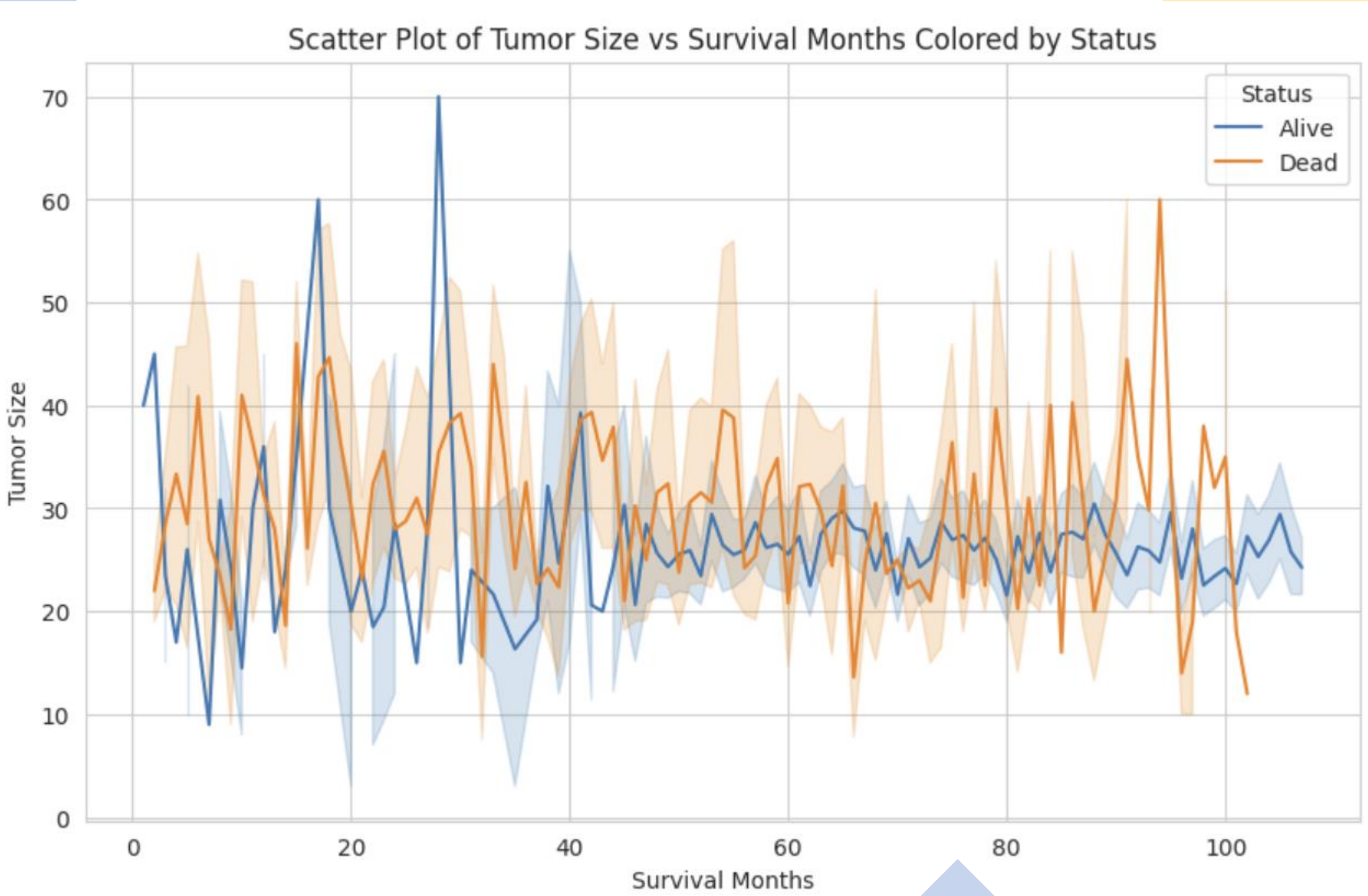

Status
Estrogen Status vs Status
Status
Alive
Dead
count
3000
2500
2000
1500
1000
500
0
Positive
Negative
Estrogen Status

```
# Visualization codes
data=df
plt.figure(figsize=(10, 6))
sns.boxplot(data=data, x='Grade', y='Tumor Size')
plt.title('Boxplot of Tumor Size for Each Grade')
plt.xlabel('Grade')
plt.ylabel('Tumor Size')
plt.show()
```



```
plt.figure(figsize=(10, 6))
sns.violinplot(data=data, x='Grade', y='Survival Months', inner='quartile')
plt.title('Violin Plot for Survival Months Across Different Grades')
plt.xlabel('Grade')
plt.ylabel('Survival Months')
plt.show()
```




Scatter Plot of Tumor Size vs Survival Months Colored by Status
Status
Alive
Dead
Tumor Size
Survival Months
0
10
20
30
40
50
60
70
0
20
40
60
80
100

STATISTICAL TESTS

NORMALITY TEST – SHAPIRO TEST

- The assessment of data distribution involved employing the Shapiro-Wilk test individually for each column.
- The results uniformly revealed p-values less than the threshold of 0.05, signifying a substantial departure from a normal distribution in all variables tested.
- Consequently, we proceeded with non-parametric tests due to the not-normal distribution observed.

```
Statistics=0.976, p=0.000
Sample does not look Gaussian (not normally distributed)
Statistics=0.436, p=0.000
Sample does not look Gaussian (not normally distributed)
Statistics=0.724, p=0.000
Sample does not look Gaussian (not normally distributed)
Statistics=0.804, p=0.000
Sample does not look Gaussian (not normally distributed)
Statistics=0.641, p=0.000
Sample does not look Gaussian (not normally distributed)
Statistics=0.835, p=0.000
Sample does not look Gaussian (not normally distributed)
Statistics=0.666, p=0.000
Sample does not look Gaussian (not normally distributed)
Statistics=0.788, p=0.000
Sample does not look Gaussian (not normally distributed)
Statistics=0.132, p=0.000
Sample does not look Gaussian (not normally distributed)
Statistics=0.963, p=0.000
Sample does not look Gaussian (not normally distributed)
Statistics=0.269, p=0.000
Sample does not look Gaussian (not normally distributed)
Statistics=0.458, p=0.000
Sample does not look Gaussian (not normally distributed)
Statistics=0.960, p=0.000
Sample does not look Gaussian (not normally distributed)
Statistics=0.653, p=0.000
Sample does not look Gaussian (not normally distributed)
```

```python
from scipy.stats import shapiro
data = df
for column in data.columns:
    if pd.api.types.is_numeric_dtype(data[column]):
        stat, p = shapiro(data[column].dropna())
        print('Statistics=%.3f, p=%.3f' % (stat, p))
        if p > 0.05:
            print('Sample looks Gaussian (normally distributed)')
        else:
            print('Sample does not look Gaussian (not normally distributed)')
```

CHI-SQUARE TEST

- The Chi-Square test revealed strong links between 'Survival Months' and crucial factors. Disease stage and hormone status ('6th Stage,' 'N Stage,' 'Estrogen Status,' 'Progesterone Status'), Regional Node Positive had a big impact. Other things like 'Differentiate,' 'Grade,' and 'T Stage' also mattered, but less. Factors like 'Race,' 'Tumor Size,' and 'Age' showed different levels of influence on how long someone survives."

- The Chi-Square test findings showcased significant relationships between 'Survival Months' and the variables investigated, culminating in a clear rejection of the null hypothesis

```
{'Survival Months': 2.9183813814777 59e-256,
 '6th Stage': 9.830332296203994e-60,
 'N Stage': 2.430140625217663e-59,
 'Reginol Node Positive': 2.292691013454 1016e-51,
 'Estrogen Status': 3.0526081181489177e-31,
 'Progesterone Status': 5.392079685518964e-29,
 'differentiate': 3.0913516733336542e-24,
 'Grade': 3.0913516733336542e-24,
 'T Stage ': 2.7790953099786567e-22,
 'A Stage': 2.2264262284984456e-09,
 'Race': 8.440928800112451e-07,
 'Tumor Size': 3.162931073136348e-06,
 'Marital Status': 1.1027694804532703e-05,
 'Age': 1.880215544112303e-05,
 'Regional Node Examined': 0.04501058878879 1626}
```

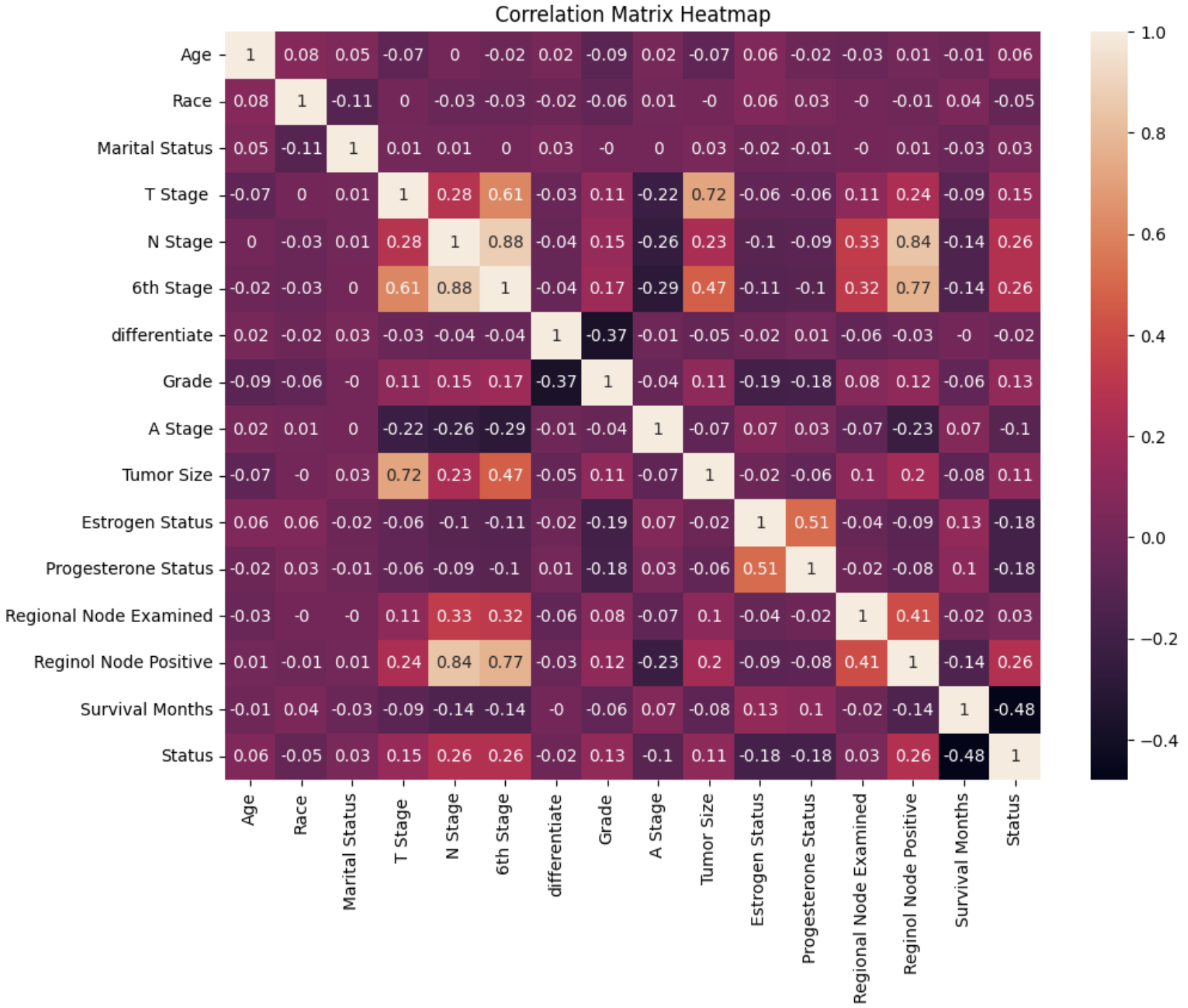
KENDALL TAU TEST

The Kendall Tau test shows that Status, 6th Stage ,N Stage ,Regional Node Positive, Estrogen Status are significantly linked to 'Survival Months,' indicating clear connections between these factors by rejecting the null hypothesis.

```python
# List of categorical columns
categorical_columns = ['Age', 'Race', 'Marital Status', 'T Stage ', 'N Stage', '6th Stage',
                       'differentiate', 'Grade', 'A Stage', 'Tumor Size', 'Estrogen Status',
                       'Progesterone Status', 'Regional Node Examined',
                       'Reginol Node Positive', 'Status']

# Dictionary to store correlation coefficients and p-values
kendall_stats = {}

for col in categorical_columns:

  # Calculate Kendall's tau
  corr, p_value = kendalltau(df[col], df['Survival Months'])

  # Store coefficient and p-value
  kendall_stats[col] = (abs(corr), p_value)
```

```
tus: Kendall Tau = 0.3414476058954227, p-value = 5.646568994090724e-153
 Stage: Kendall Tau = 0.08809000624467192, p-value = 1.1960892764624344e-1
tage: Kendall Tau = 0.08720799856359678, p-value = 3.2438024562915286e-12
inol Node Positive: Kendall Tau = 0.0803490400570991, p-value = 2.23317184
rogen Status: Kendall Tau = 0.07197890836008296, p-value = 2.7928137010130
tage : Kendall Tau = 0.05924776795136345, p-value = 1.4801486674019642e-06
or Size: Kendall Tau = 0.05281073860255156, p-value = 1.2629377509947379e-
gesterone Status: Kendall Tau = 0.052621500973826264, p-value = 4.90057477
tage: Kendall Tau = 0.04517465111510899, p-value = 0.0004908259976979512
de: Kendall Tau = 0.03839540570452738, p-value = 0.001989844017965658
ferentiate: Kendall Tau = 0.019348945389621706, p-value = 0.11922125325907
e: Kendall Tau = 0.01626957857251364, p-value = 0.2009803297151208
: Kendall Tau = 0.013217678645805194, p-value = 0.2187339440104471
ital Status: Kendall Tau = 0.012480198825198931, p-value = 0.3099101134537
ional Node Examined: Kendall Tau = 0.007512517839020049, p-value = 0.48588
```

Correlation Heatmap



MODEL DEVELOPMENT & ANALYSIS

- As stated in our aim we are predicting the survival rate by using the variable ‘status’ as a target variable which is a good indicator of survival rate of a cancer patient.
- We created a train-test split of 80-20 on the data set to evaluate our machine-learning models.
- We applied the below classification models to predict survival rate:
    1. Decision tree classifier model
    2. XGboost classifier model
    3. Random forest
    4. Logistic regression
    5. KNN classifier
- We assessed the model's performance by applying it to the test set in the dataset, utilizing the classification report to examine various metrics. Additionally, we utilized the receiver operating characteristic (ROC) curve to gain insights into the model's behavior.

Creating the Target Variable and Independent Variables from Data:

```
#Aim 2
#split into X/y
X = df[['Age', 'T Stage ', 'N Stage', '6th Stage',
       'differentiate', 'Grade', 'A Stage', 'Tumor Size', 'Estrogen Status',
       'Progesterone Status']]
y = df["Status"]
```

Test Train Split for Model Evaluation:

```
# split into training and test
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size = 0.2)
print(X_train.shape)
print(X_test.shape)
```

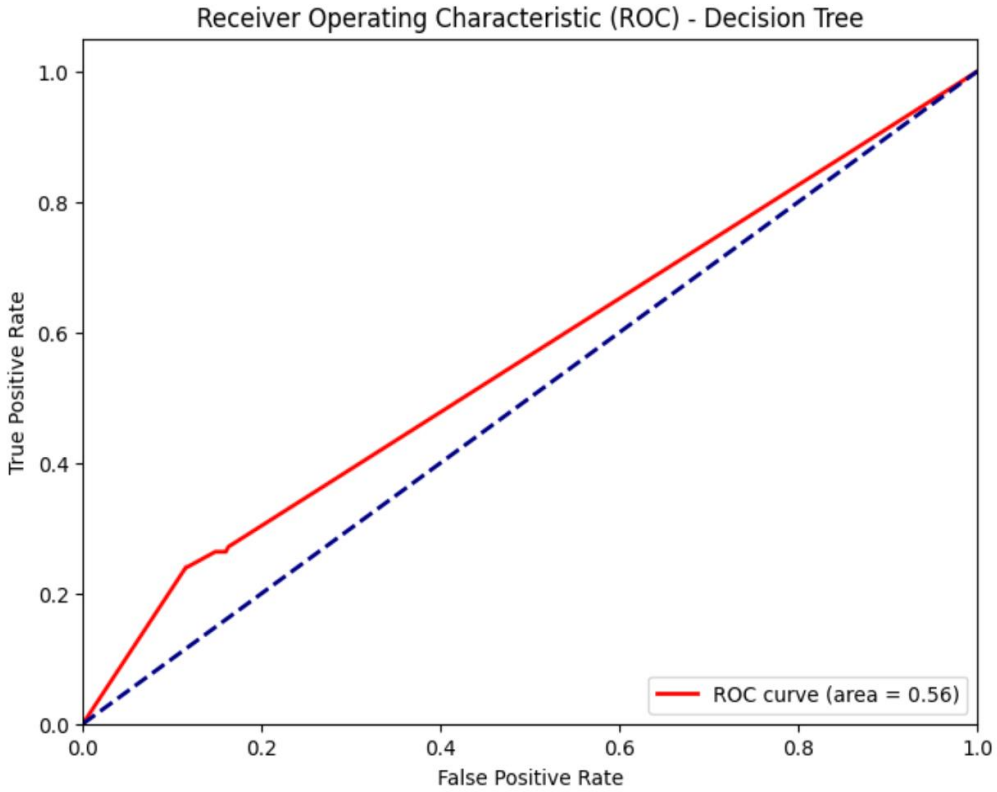
```
(3219, 10)
(805, 10)
```

- As seen above there are 10 columns in the data set as we dropped the Race, Marital Status, Regional Node Examined, Regional Node Positive, Survival Months, column because it will not have a significant impact on the model building.

DECISION TREE CLASSIFIER MODEL

Classification Report on test data:

```
Accuracy:        0.7801242236024845
              precision    recall  f1-score   support

           0       0.86      0.88      0.87       676
           1       0.28      0.24      0.26       129

    accuracy                           0.78       805
   macro avg       0.57      0.56      0.57       805
weighted avg       0.77      0.78      0.77       805

Confusin Matrix:
 [[597  79]
 [ 98  31]]
```



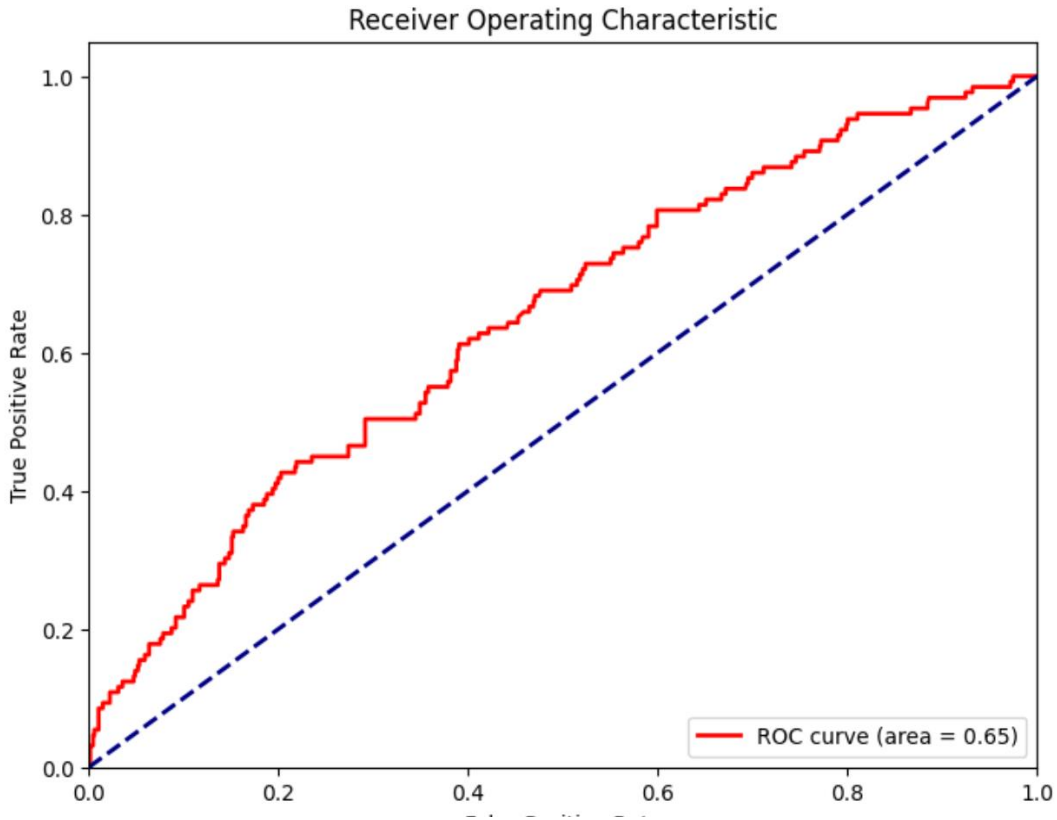
- The decision tree classifier model achieves an accuracy of 77% on the test set, which is relatively high. This means that the model can correctly classify 77% of the cases.
- The ROC curve shows that the model has a good ability to distinguish between positive and negative cases, with an AUC of 0.58.

XGBOOST CLASSIFIER MODEL

Classification Report on test data:

The model has an accuracy of 80.8%, meaning that it correctly predicts the class. of 80.8% of the data points.

```
Accuracy:         0.8248447204968944
              precision    recall  f1-score   support

           0       0.85      0.96      0.90       676
           1       0.36      0.12      0.18       129

    accuracy                           0.82       805
   macro avg       0.61      0.54      0.54       805
weighted avg       0.77      0.82      0.79       805

Confusin Matrix:
 [[648  28]
 [113  16]]
```



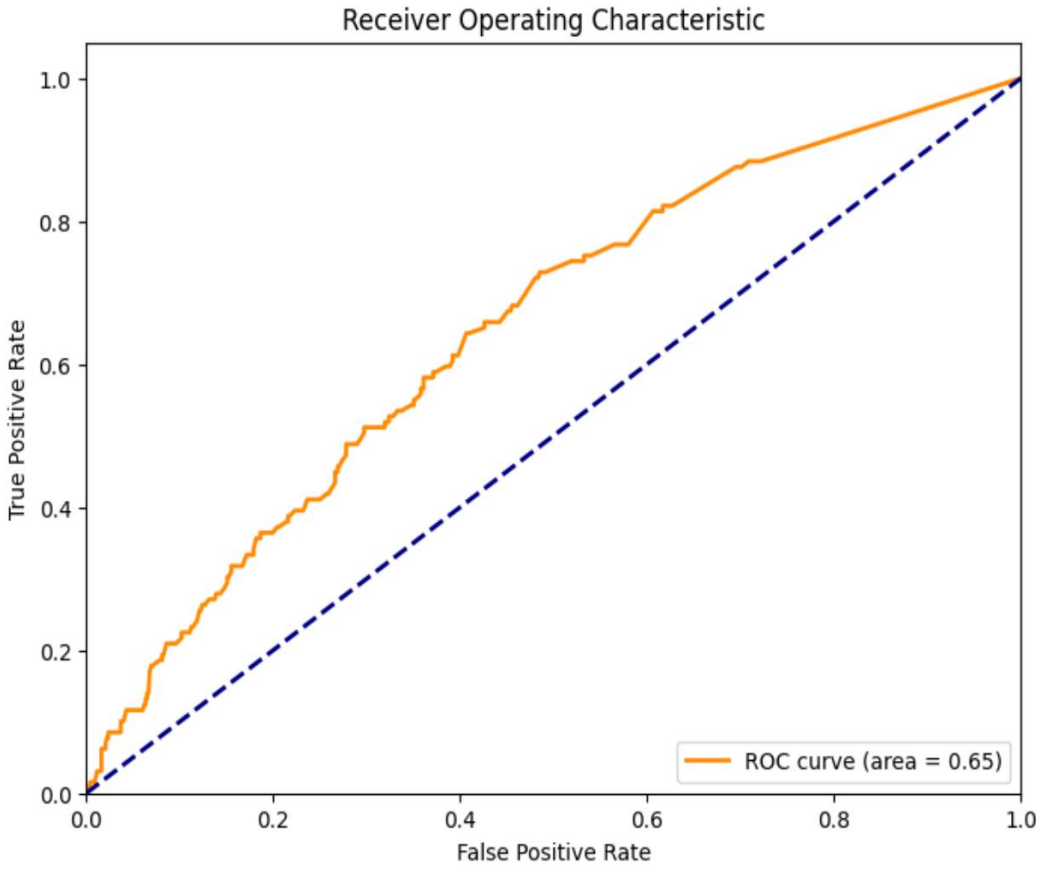
- The model has an accuracy of 80.8%, meaning that it correctly predicts the class of 80.8% of the data points.
- The ROC curve shows that the model has a high true positive rate (TPR) at all false positive rates (FPR), indicating that it is good at identifying positive cases without generating too many false positives.

RANDOM FOREST

Classification Report on test data:

We executed the Random forest model and following results are obtained.

```
Accuracy:         0.8099378881987578
              precision    recall  f1-score   support

           0       0.85      0.93      0.89       676
           1       0.32      0.17      0.22       129

    accuracy                           0.81       805
   macro avg       0.59      0.55      0.56       805
weighted avg       0.77      0.81      0.78       805

Confusin Matrix:
 [[630  46]
 [107  22]]
```



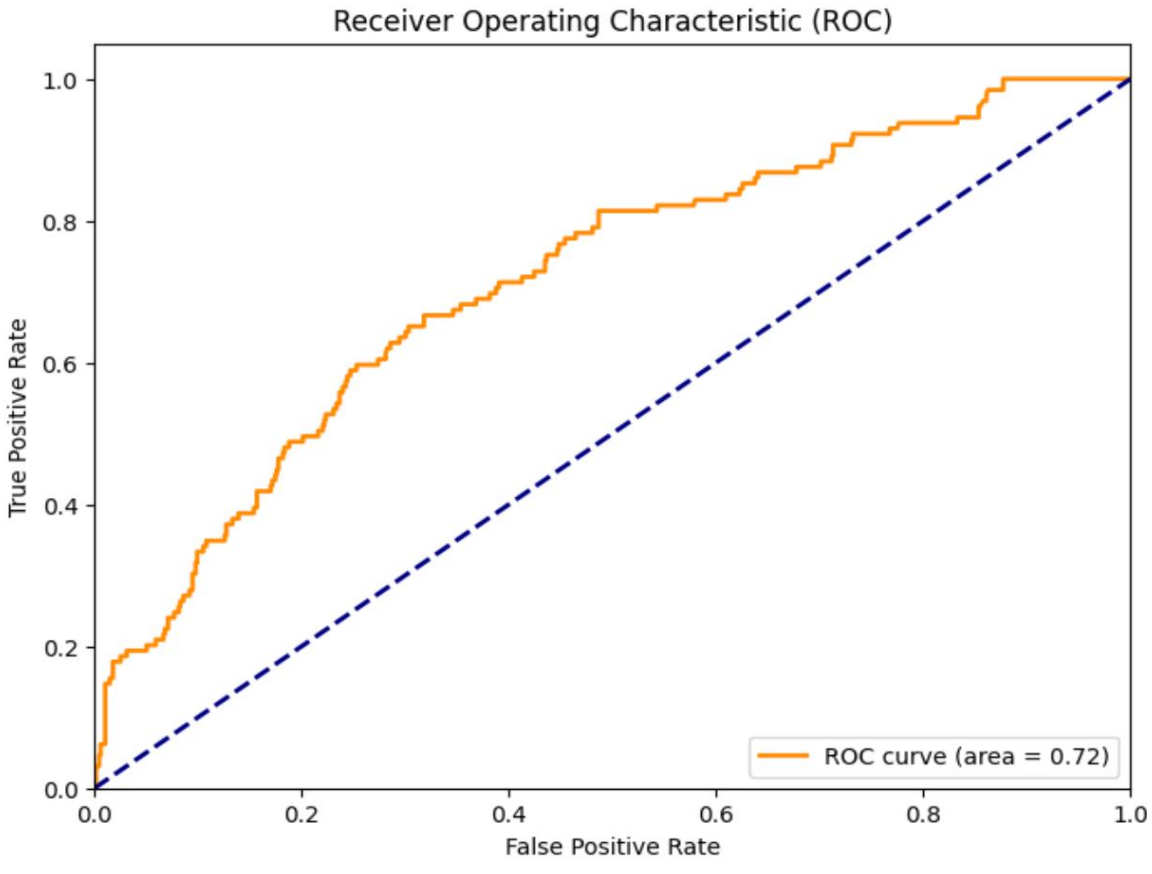
- The model achieved an accuracy of 80.2%, indicating that it correctly classified 80.2% of the data points
- Additionally, the model's ROC AUC score of 0.65 demonstrates its ability to distinguish between positive and negative cases effectively.

LOGISTIC REGRESSION

Classification Report on test data:

Following results are obtained after executing the Logistic Regression model.

```
Accuracy:        0.8521739130434782
              precision    recall  f1-score   support

           0       0.86      0.99      0.92       676
           1       0.68      0.15      0.24       129

    accuracy                           0.85       805
   macro avg       0.77      0.57      0.58       805
weighted avg       0.83      0.85      0.81       805

Confusin Matrix:
 [[667   9]
 [110  19]]
```



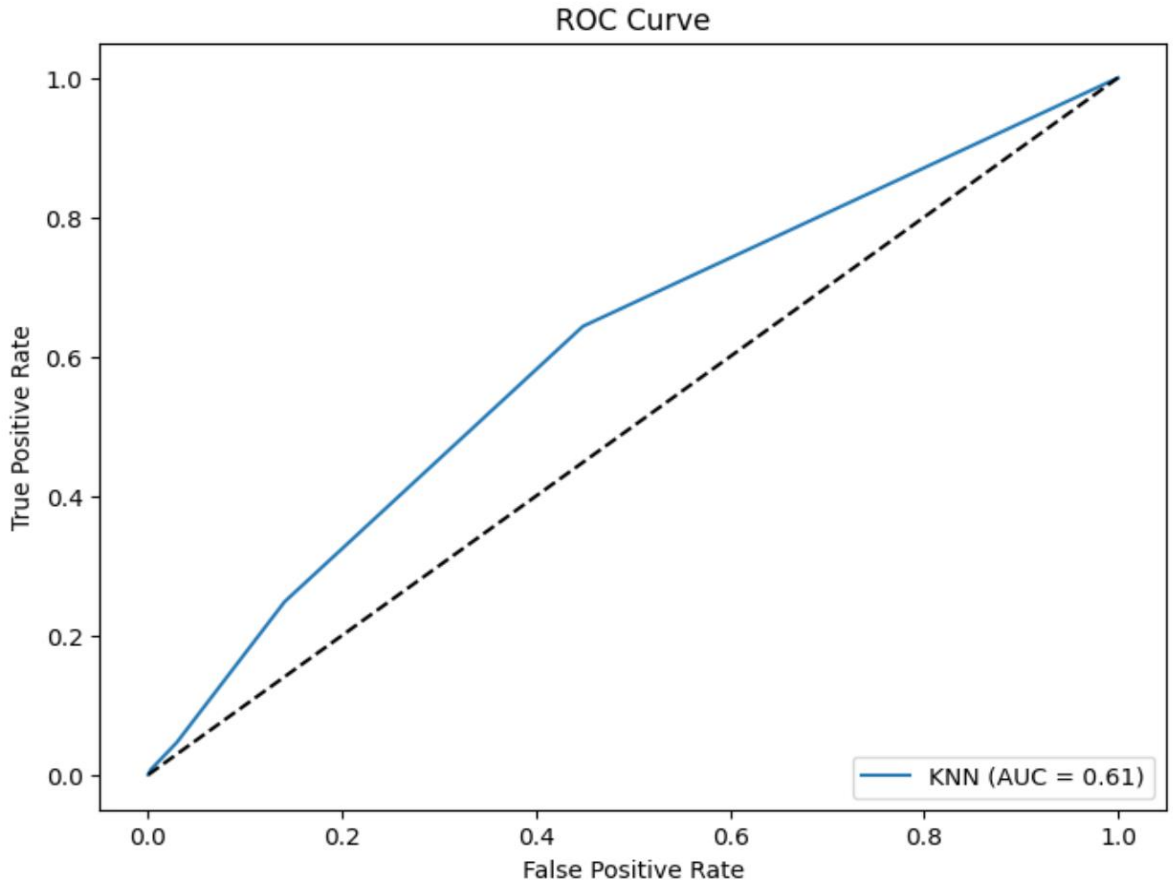
- The Logistic Regression classifier model achieved an accuracy of 84.6%, indicating that it correctly classified 84.6% of the data points in the test dataset.
- The model's ROC AUC score of 0.72 demonstrates its ability to distinguish between positive and negative cases effectively

KNN CLASSIFIER

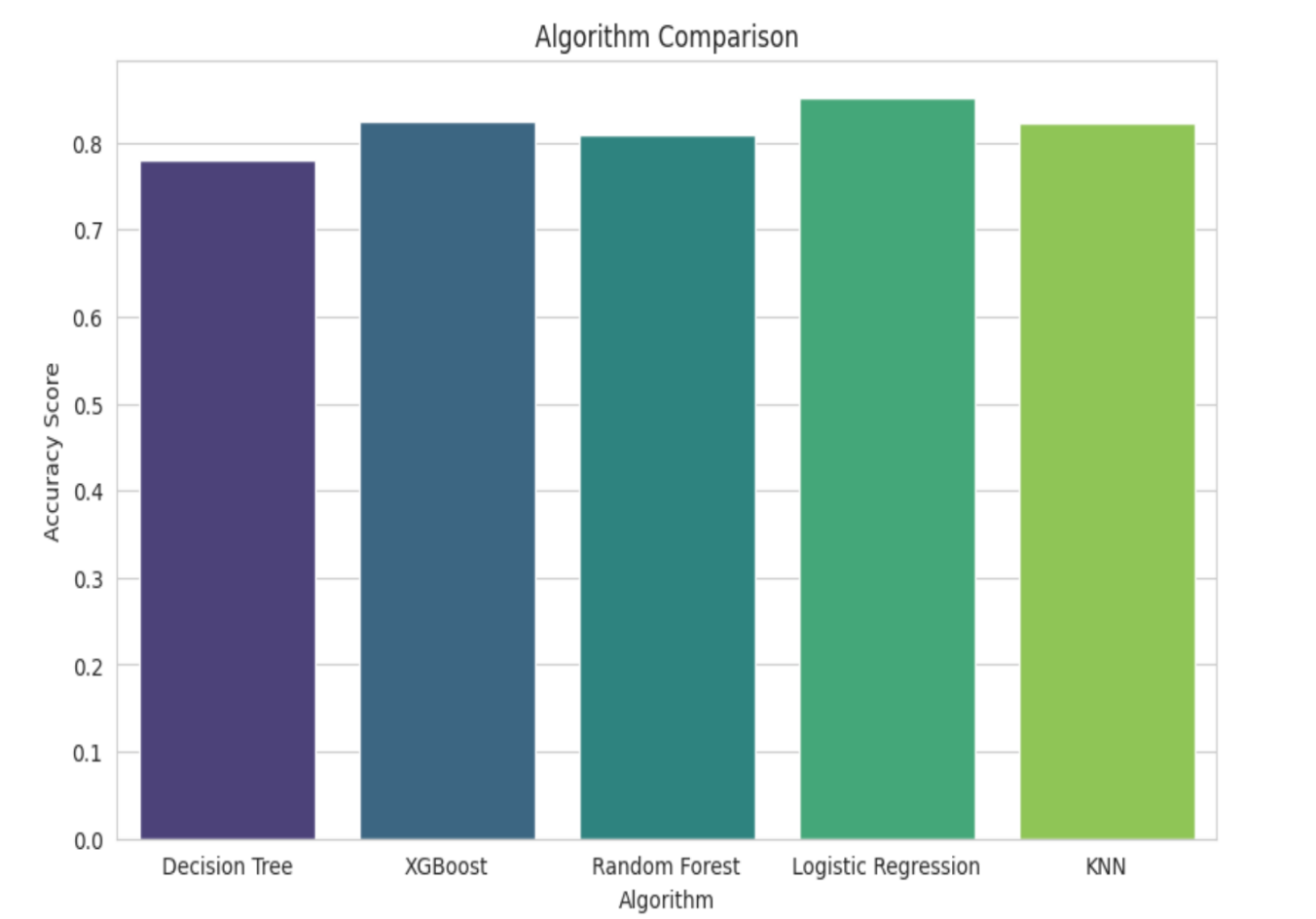
Classification Report on test data:

Following results are obtained after executing the KNN Classifier model.

```
Accuracy:        0.822360248447205
              precision    recall  f1-score   support

           0       0.84      0.97      0.90       676
           1       0.23      0.05      0.08       129

    accuracy                           0.82       805
   macro avg       0.54      0.51      0.49       805
weighted avg       0.74      0.82      0.77       805

Confusin Matrix:
 [[656  20]
 [123   6]]
```



- The KNN classifier model achieved an accuracy of 82.5% on the test dataset, indicating that it correctly classified 82.5% of the data points.
- The model's ROC AUC score of 0.61 demonstrates its ability to distinguish between positive and negative cases effectively.



- By comparing all the models, the Logistic Regression model is the best model for predicting the target variable. It has the highest accuracy score and can generalize well to new data.

Research Questions

Are there any significant interactions or correlations among the various clinical and biological features studied that can further enhance the accuracy of predicting breast cancer survival status?

Yes, the factors such as Regional Node Positive , VI Stage, N Stage, Estrogen Status, Progesterone Status , Status has a significant impact on breast cancer and survival status..

References:

- Breast_cancer. (n.d.). www.kaggle.com. https://www.kaggle.com/datasets/alaahussien/breast-cancer

- De Sanctis, V., Soliman, A. T., Daar, S., Tzoulis, P., Fiscina, B., Kattamis, C., & International Network Of Clinicians For Endocrinopathies In Thalassemia And Adolescence Medicine Icet-A. (2022). Retrospective observational studies: Lights and shadows for medical writers. Acta Bio-Medica : Atenei Parmensis, 93(5), e2022319. https://doi.org/10.23750/abm.v93i5.13179

- Feng, Y., Spezia, M., Huang, S., Yuan, C., Zeng, Z., Zhang, L., Ji, X., Liu, W., Huang, B., Luo, W., Liu, B., Lei, Y., Du, S., Vuppalapati, A., Luu, H. H., Haydon, R. C., He, T.-C., & Ren, G. (2018). Breast cancer development and progression: Risk factors, cancer stem cells, signaling pathways, genomics, and molecular pathogenesis. Genes & Diseases, 5(2), 77–106. https://doi.org/10.1016/j.gendis.2018.05.001

- Satpathi, S., Gaurkar, S. S., Potdukhe, A., & Wanjari, M. B. (2023c). Unveiling the Role of Hormonal Imbalance in Breast Cancer Development: A Comprehensive Review. *Cureus*, *15*(7), e41737. https://doi.org/10.7759/cureus.41737

APPENDIX

```
df = pd.read_csv("Breast_Cancer (1).csv")
print(df.shape)
df.head
```

```
(4024, 16)
<bound method NDFrame.head of       Age   Race Marital Status T Stage  N Stage 6th Stage  \
0      68  White        Married       T1      N1       IIA
1      50  White        Married       T2      N2      IIIA
2      58  White       Divorced       T3      N3      IIIC
3      58  White        Married       T1      N1       IIA
4      47  White        Married       T2      N1       IIB
...   ...    ...            ...      ...     ...       ...
4019   62  Other        Married       T1      N1       IIA
4020   56  White       Divorced       T2      N2      IIIA
4021   68  White        Married       T2      N1       IIB
4022   58  Black       Divorced       T2      N1       IIB
4023   46  White        Married       T2      N1       IIB

                differentiate Grade   A Stage  Tumor Size Estrogen Status  \
0         Poorly differentiated     3  Regional           4        Positive
1     Moderately differentiated     2  Regional          35        Positive
2     Moderately differentiated     2  Regional          63        Positive
3         Poorly differentiated     3  Regional          18        Positive
4         Poorly differentiated     3  Regional          41        Positive
...                         ...   ...       ...         ...             ...
4019  Moderately differentiated     2  Regional           9        Positive
4020  Moderately differentiated     2  Regional          46        Positive
4021  Moderately differentiated     2  Regional          22        Positive
4022  Moderately differentiated     2  Regional          44        Positive
4023  Moderately differentiated     2  Regional          30        Positive
```

```
df.describe()
```

	Age	Tumor Size	Regional Node Examined	Reginol Node Positive	Survival Months
count	4024.000000	4024.000000	4024.000000	4024.000000	4024.000000
mean	53.972167	30.473658	14.357107	4.158052	71.297962
std	8.963134	21.119696	8.099675	5.109331	22.921430
min	30.000000	1.000000	1.000000	1.000000	1.000000
25%	47.000000	16.000000	9.000000	1.000000	56.000000
50%	54.000000	25.000000	14.000000	2.000000	73.000000
75%	61.000000	38.000000	19.000000	5.000000	90.000000
max	69.000000	140.000000	61.000000	46.000000	107.000000

```python
# Find duplicated rows
duplicate_rows = df[df.duplicated(keep=False)]

# Count the occurrences of each duplicated row
duplicate_counts = duplicate_rows.groupby(list(df.columns)).size().reset
print(duplicate_rows)
# Eliminate duplicated rows
df_no_duplicates = df.drop_duplicates()
```

```
     Age   Race Marital Status T Stage  N Stage 6th Stage  \
213   63  White        Married       T1      N1       IIA
436   63  White        Married       T1      N1       IIA

                differentiate Grade   A Stage  Tumor Size Estrogen Stat
213  Moderately differentiated     2  Regional          17        Positi
436  Moderately differentiated     2  Regional          17        Positi

    Progesterone Status  Regional Node Examined  Reginol Node Positive
213            Positive                       9                      1
436            Positive                       9                      1

     Survival Months Status
213               56  Alive
436               56  Alive
```

```python
# Number of unique columns
uni_col = df.columns.nunique()

# Number of unique rows
uni_row = df.nunique(axis=0)

print(f"Number of unique columns: {uni_col}")
print(f"Number of unique rows: {uni_row}")
```

```
Number of unique columns: 16
Number of unique rows: Age                       40
Race                          3
Marital Status                5
T Stage                       4
N Stage                       3
6th Stage                     5
differentiate                 4
Grade                         4
A Stage                       2
Tumor Size                  110
Estrogen Status               2
Progesterone Status           2
Regional Node Examined       54
Reginol Node Positive        38
Survival Months             107
Status                        2
dtype: int64
```

```python
import pandas as pd

# Load your DataFrame
df = pd.read_csv('Breast_Cancer (1).csv')  # Replace with your fil

# Calculate IQR for 'Tumor Size'
Q1 = df['Tumor Size'].quantile(0.25)
Q3 = df['Tumor Size'].quantile(0.75)
IQR = Q3 - Q1

# Define bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Replace outliers with the median value
median_value = df['Tumor Size'].median()
#df.loc[df['Tumor Size'] < lower_bound, 'Tumor Size'] = median_val
df.loc[df['Tumor Size'] > upper_bound, 'Tumor Size'] = median_valu

# Save the cleaned DataFrame, if needed
df.to_csv('b.csv', index=False)
```

```python
### ----------  ENCODING  ----------

import pandas as pd
from sklearn.preprocessing import LabelEncoder

# Assuming df as DataFrame

# List of columns to encode
columns_to_encode = ['T Stage ', '6th Stage', 'N Stage', 'Race', 'differentiate', 'Marital Status', 'Grade', 'A Stage','Estrogen Status','Prog

# Initialize LabelEncoder
label_encoder = LabelEncoder()

# Apply LabelEncoder to each column
for col in columns_to_encode:
    df[col] = label_encoder.fit_transform(df[col])

# Display the first few rows of the updated DataFrame
print(df.head())
```

```python
from scipy.stats import shapiro
data = df
for column in data.columns:
    if pd.api.types.is_numeric_dtype(data[column]):
        stat, p = shapiro(data[column].dropna())
        print('Statistics=%.3f, p=%.3f' % (stat, p))
        if p > 0.05:
            print('Sample looks Gaussian (normally distributed)')
        else:
            print('Sample does not look Gaussian (not normally distributed)')
```

```
Statistics=0.976, p=0.000
Sample does not look Gaussian (not normally distributed)
Statistics=0.436, p=0.000
Sample does not look Gaussian (not normally distributed)
Statistics=0.724, p=0.000
Sample does not look Gaussian (not normally distributed)
Statistics=0.804, p=0.000
Sample does not look Gaussian (not normally distributed)
Statistics=0.641, p=0.000
Sample does not look Gaussian (not normally distributed)
Statistics=0.835, p=0.000
Sample does not look Gaussian (not normally distributed)
Statistics=0.666, p=0.000
Sample does not look Gaussian (not normally distributed)
Statistics=0.788, p=0.000
Sample does not look Gaussian (not normally distributed)
Statistics=0.132, p=0.000
Sample does not look Gaussian (not normally distributed)
```

```python
# perform chi- square test for feature importance on Status

import pandas as pd
from scipy.stats import chi2_contingency


# Assuming df is your DataFrame and 'Status' is your target variable

# List of categorical columns (excluding 'Status')
categorical_columns = ['Age', 'Race', 'Marital Status', 'T Stage ', 'N Stage', '6th Stage',
       'differentiate', 'Grade', 'A Stage', 'Tumor Size', 'Estrogen Status',
       'Progesterone Status', 'Regional Node Examined',
       'Reginol Node Positive', 'Survival Months']

# Dictionary to hold p-values
p_values = {}

for col in categorical_columns:
    # Creating a contingency table
    contingency_table = pd.crosstab(df[col], df['Status'])

    # Performing the Chi-square test
    chi2, p, dof, ex = chi2_contingency(contingency_table)

    # Storing the p-value for the feature
    p_values[col] = p

# Sorting features by their p-value
sorted_p_values = {k: v for k, v in sorted(p_values.items(), key=lambda item: item[1])}

# Display sorted features by importance
sorted_p_values
```

```python
from scipy.stats import kendalltau

# List of categorical columns
categorical_columns = ['Age', 'Race', 'Marital Status', 'T Stage ', 'N Stage', '6th Stage',
                       'differentiate', 'Grade', 'A Stage', 'Tumor Size', 'Estrogen Status',
                       'Progesterone Status', 'Regional Node Examined',
                       'Reginol Node Positive', 'Status']

# Dictionary to store correlation coefficients and p-values
kendall_stats = {}

for col in categorical_columns:

  # Calculate Kendall's tau
  corr, p_value = kendalltau(df[col], df['Survival Months'])

  # Store coefficient and p-value
  kendall_stats[col] = (abs(corr), p_value)

# Sort features by correlation
sorted_stats = {k: v for k, v in sorted(kendall_stats.items(), key=lambda item: item[1][0], reverse=True)}

# Print correlations and p-values
for feat, (corr, p) in sorted_stats.items():
  print(f'{feat}: Kendall Tau = {corr}, p-value = {p}')
```

```python
##----------------    DECISION TREE CLASSIFIER MODEL -------------

import seaborn as sns
from sklearn.metrics import confusion_matrix
columns_consider= ['Age', 'T Stage ', 'N Stage', '6th Stage',
       'differentiate', 'Grade', 'A Stage', 'Tumor Size', 'Estrogen Status',
       'Progesterone Status']
# Model building using DECISION TREE
dtc = DecisionTreeClassifier(random_state = 606)
dtc.fit(X_train[columns_consider], y_train)
y_pred_dt= dtc.predict(X_test[columns_consider])

# calculating Accuracy
print("Accuracy: \t",accuracy_score(y_test, y_pred_dt))
print(classification_report(y_test, y_pred_dt))
print("Confusin Matrix: \n",confusion_matrix(y_test, y_pred_dt))

# Visual Representation of confusion matrix

cm = confusion_matrix(y_test, y_pred_dt)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=["Class 0", "Class 1"], yticklabels=["Class 0", "Class 1"])
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()


# ROC for decision tree classifier
dtc = DecisionTreeClassifier(random_state=606)
dtc.fit(X_train[columns_consider], y_train)

y_scores_dt = dtc.predict_proba(X_test[columns_consider])[:, 1]

fpr, tpr, _ = roc_curve(y_test, y_scores_dt)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='red', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
```

```
## ------------------------  XBBOOST CLASSIFIER MODEL  ------------------------------

# Model building using XGBOOST
columns_required=columns_consider
xgbc = xgb.XGBClassifier(random_state = 606)
xgbc.fit(X_train[columns_required], y_train)
y_pred_xg= xgbc.predict(X_test[columns_required])

#calculating Accuracy
print("Accuracy: \t",accuracy_score(y_test, y_pred_xg))
print(classification_report(y_test, y_pred_xg))
print("Confusin Matrix: \n",confusion_matrix(y_test, y_pred_xg))

# Visual Representation of confusion matrix

cm = confusion_matrix(y_test, y_pred_dt)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=["Class 0", "Class 1"], yticklabels=["Class 0", "Class 1"])
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()

# ROC for XG Boost classifier
y_pred_xg = xgbc.predict_proba(X_test[columns_required])[:, 1]
fpr, tpr, _ = roc_curve(y_test, y_pred_xg)
roc_auc = auc(fpr, tpr)

# Plotting the ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='red', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()
```

```
# Model building using RANDOMFOREST
rfc = RandomForestClassifier(random_state = 606)
rfc.fit(X_train[columns_required], y_train)
y_pred= rfc.predict(X_test[columns_required])
print("Accuracy: \t",accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
print("Confusin Matrix: \n",confusion_matrix(y_test, y_pred))

# Visual Representation of confusion matrix

cm = confusion_matrix(y_test, y_pred_dt)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=["Class 0", "Class 1"], yticklabels=["Class 0", "Class 1"])
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()

# ROC for Random Forest Classifier

rfc = RandomForestClassifier(random_state = 606)
rfc.fit(X_train[columns_required], y_train)
y_pred = rfc.predict(X_test[columns_required])

y_pred_prob = rfc.predict_proba(X_test[columns_required])[:, 1]
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()
```

```
# Model building using Logistic Regression
lrc = LogisticRegression()
lrc.fit(X_train[columns_required], y_train)
y_pred_lr= lrc.predict(X_test[columns_required])


#calculating Accuracy
print("Accuracy: \t",accuracy_score(y_test, y_pred_lr))
print(classification_report(y_test, y_pred_lr))
print("Confusin Matrix: \n",confusion_matrix(y_test, y_pred_lr))

# Visual Representation of confusion matrix

cm = confusion_matrix(y_test, y_pred_dt)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=["Class 0", "Class 1"], yticklabels=["Class 0", "Class 1"])
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()

# ROC FOR LOGISTIC REGRESSION

y_pred_prob = lrc.predict_proba(X_test[columns_required])[:, 1]

fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)
roc_auc = auc(fpr, tpr)

# Plotting the ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC)')
plt.legend(loc="lower right")
```

```
# Model building using KNN Classifier
knc = KNeighborsClassifier( metric='minkowski')
knc.fit(X_train[columns_required], y_train)
y_pred_knc= knc.predict(X_test[columns_required])
#Calculating Accuracy
print("Accuracy: \t",accuracy_score(y_test, y_pred_knc))
print(classification_report(y_test, y_pred_knc))
print("Confusin Matrix: \n",confusion_matrix(y_test, y_pred_knc))

# Visual Representation of confusion matrix

cm = confusion_matrix(y_test, y_pred_dt)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=["Class 0", "Class 1"], yticklabels=["Class 0", "Class 1"])
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()

y_pred_prob_knc = knc.predict_proba(X_test[columns_required])[:, 1]

# Calculate ROC Curve
fpr_knc, tpr_knc, thresholds_knc = roc_curve(y_test, y_pred_prob_knc)

# Calculate AUC (Area under the ROC Curve)
auc_knc = auc(fpr_knc, tpr_knc)

# Plotting ROC Curve
plt.figure(figsize=(8, 6))
plt.plot(fpr_knc, tpr_knc, label=f'KNN (AUC = {auc_knc:.2f})')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc='lower right')
plt.show()
```

```
### ---------- Age Distribution of Breast Cancer Deceased ----------------

# Assuming your DataFrame is named 'data', 'Age' is the age column, and 'Status' is the status column
# Let's assume '1' in 'Status' indicates death

# Filter the DataFrame for individuals who died
deceased = df[df['Status'] == 1]

plt.figure(figsize=(10, 6))

# Create a histogram for the 'Age' of deceased individuals
sns.histplot(deceased['Age'], bins=20, kde=False, color='red')

# Add title and labels
plt.title('Age Distribution of Deceased Individuals')
plt.xlabel('Age')
plt.ylabel('Frequency')

# Display the plot
plt.show()
```

```python
df = pd.read_csv("Breast_Cancer (1).csv")
numeric_cols = df.select_dtypes(include=['float64', 'int64']).columns
categorical_cols = df.select_dtypes(include=['object', 'category']).columns

# Assuming 'Status' as target column
target_col = 'Status'

# Define the number of rows needed for subplots
num_rows = len(df.columns) - 1  # excluding the target column

# Create subplots
fig, axes = plt.subplots(nrows=num_rows, ncols=1, figsize=(10, 5 * num_rows))

# Iterate through columns and create plots
for i, column in enumerate(df.columns):
    if column == target_col:
        continue  # Skip the target column

    ax = axes[i] if num_rows > 1 else axes  # Handle case of single subplot

    if column in numeric_cols:
        # For numeric columns, use a boxplot or scatterplot
        sns.boxplot(x=target_col, y=column, data=df, ax=ax)


    elif column in categorical_cols:
        # For categorical columns, use a countplot or barplot
        sns.countplot(x=column, hue=target_col, data=df, ax=ax)
        # Alternatively, for barplot: sns.barplot(x=target_col, y=column, data=df, ax=ax)

    ax.set_title(f'{column} vs {target_col}')

plt.tight_layout()
plt.show()
```

```python
# Visualization codes
data=df
plt.figure(figsize=(10, 6))
sns.boxplot(data=data, x='Grade', y='Tumor Size')
plt.title('Boxplot of Tumor Size for Each Grade')
plt.xlabel('Grade')
plt.ylabel('Tumor Size')
plt.show()
```

```python
# Define the number of rows needed for subplots
num_rows = len(df.columns) - 1  # excluding the target column

# Create subplots
fig, axes = plt.subplots(nrows=num_rows, ncols=1, figsize=(10, 5 * num_rows))

# Iterate through columns and create plots
for i, column in enumerate(df.columns):
    if column == target_col:
        continue  # Skip the target column

    ax = axes[i] if num_rows > 1 else axes  # Handle case of single subplot

    if column in numeric_cols:
        # For numeric columns, use a boxplot or scatterplot
        sns.boxplot(x=target_col, y=column, data=df, ax=ax)


    elif column in categorical_cols:
        # For categorical columns, use a countplot or barplot
        sns.countplot(x=column, hue=target_col, data=df, ax=ax)
        # Alternatively, for barplot: sns.barplot(x=target_col, y=column, data=df, ax=ax)
```

```
plt.figure(figsize=(10, 6))
sns.lineplot(data=data, x='Survival Months', y='Tumor Size', hue='Status')
plt.title('Scatter Plot of Tumor Size vs Survival Months Colored by Status')
plt.xlabel('Survival Months')
plt.ylabel('Tumor Size')
plt.show()
```

```
plt.figure(figsize=(10, 6))
sns.violinplot(data=data, x='Grade', y='Survival Months', inner='quartile')
plt.title('Violin Plot for Survival Months Across Different Grades')
plt.xlabel('Grade')
plt.ylabel('Survival Months')
plt.show()
```

```
plt.figure(figsize=(10, 6))
sns.barplot(data=data, x='Status', y='Tumor Size', ci=None)
plt.title('Average Tumor Size for Each Status')
plt.xlabel('Status')
plt.ylabel('Average Tumor Size')
plt.show()
```

Thank you