Survival Detection of Breast cancer using breast cancer Clinical data: REPORT

Necessary libraries:

The machine learning project leveraged a comprehensive suite of tools and libraries to facilitate data analysis, visualization, modeling, and evaluation. The project utilized fundamental libraries including NumPy, pandas, and SciPy for data manipulation, statistical analysis, and scientific computing. Visualization was enhanced through Matplotlib and Seaborn, allowing for the creation of insightful plots and charts to understand data distributions and relationships. Yellowbrick was employed for visualizing model performance and decision boundaries. Statistical tests and metrics were computed using functions from SciPy and scikit-learn, enabling rigorous evaluation of model efficacy. Modeling was executed through a variety of algorithms available in scikit-learn, such as Logistic Regression, Decision Trees, Random Forests, Support Vector Machines, K-Nearest Neighbors, and Gradient Boosting Classifiers. Additionally, advanced techniques including XGBoost, LightGBM, AdaBoost, Gaussian Naive Bayes, and Multi-layer Perceptron were explored for model optimization. Imputation of missing values and standardization of features were accomplished using scikit-learn preprocessing modules. Throughout the project, visualizations were enhanced using custom color palettes, contributing to clearer and more engaging presentations of the findings.

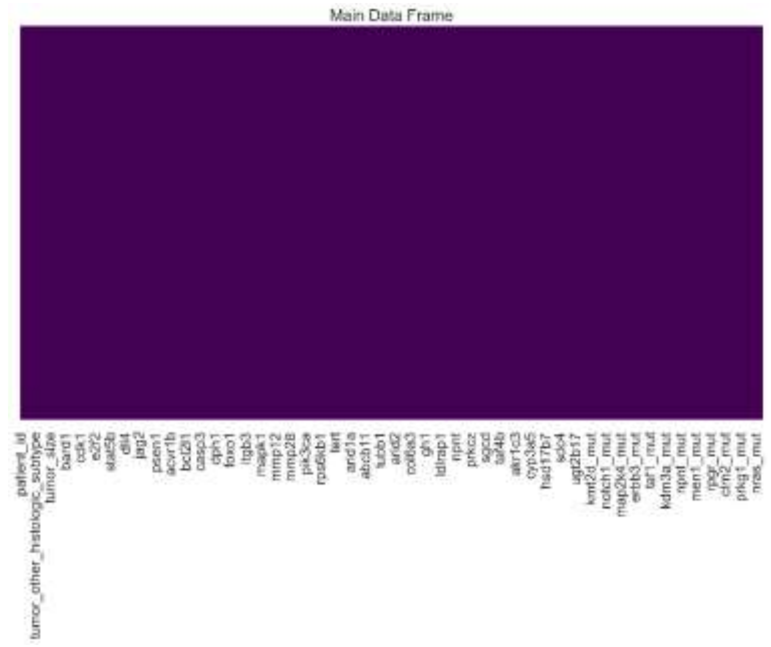
Data Description:

The dataset, sourced from 'METABRIC_RNA_Mutation.csv', comprises 1904 entries across 693 columns, with a majority of columns being of float64 type (498 columns), followed by int64 (5 columns) and object (190 columns) types. Initial examination reveals missing data present in several columns, with 'tumor_stage' exhibiting the highest count of missing values (501 entries), accounting for approximately 26.31% of its data. Similarly, '3-gene_classifier_subtype' and 'primary_tumor_laterality' possess 204 (10.71%) and 106 (5.57%) missing entries, respectively. Further columns such as 'neoplasm_histologic_grade', 'cellularity', and 'mutation_count' demonstrate varying degrees of missing data, emphasizing the necessity of data cleaning and imputation strategies. The lowest occurrence of missing values is observed in the 'ar' column, with no missing entries. Understanding and addressing these missing data points will be pivotal in ensuring the integrity and reliability of subsequent analyses and model development.

Handling missing values:

After applying a forward-fill imputation method to handle missing values, the dataset was examined again. The visualization, as shown in the heatmap, indicates successful imputation, as there are no longer any missing values present in the dataset. This process ensures that the dataset is complete and ready for subsequent analysis and modeling tasks. Here's a summary of the missing data before and after imputation. Forward-fill imputation, also known as "last observation carried forward" (LOCF), is a method used to handle missing data in a dataset. In this approach, missing values are replaced with the most recent known value in the same column.

The heatmap visualization demonstrates the successful handling of missing values, ensuring the integrity and completeness of the dataset for subsequent analyses.

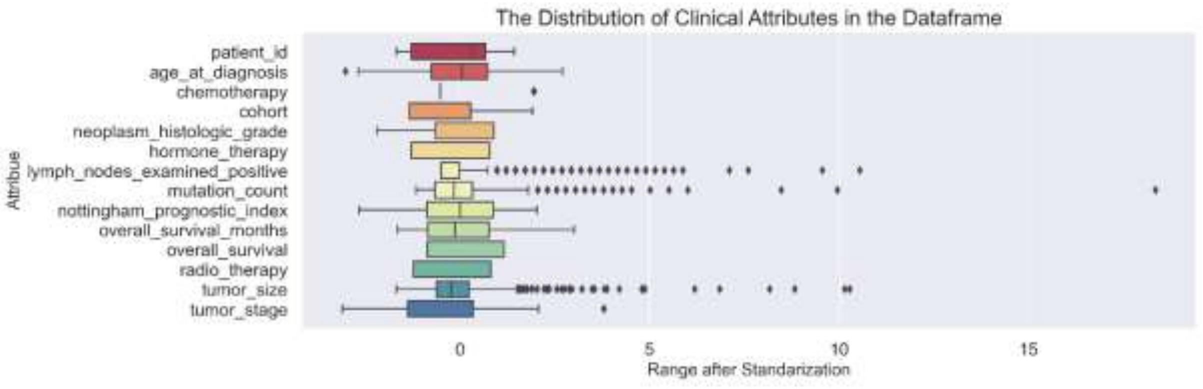


Creating new data frame for clinical attributes:

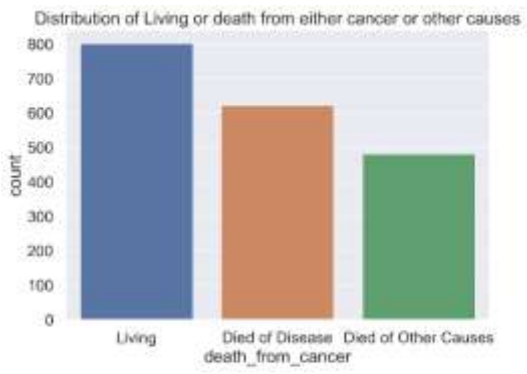
After creating a new dataframe containing only clinical attributes, the dataset comprises 1904 entries and 31 columns. The clinical features include patient demographics, tumor characteristics, treatment details, and survival outcomes.

Handling outliers:

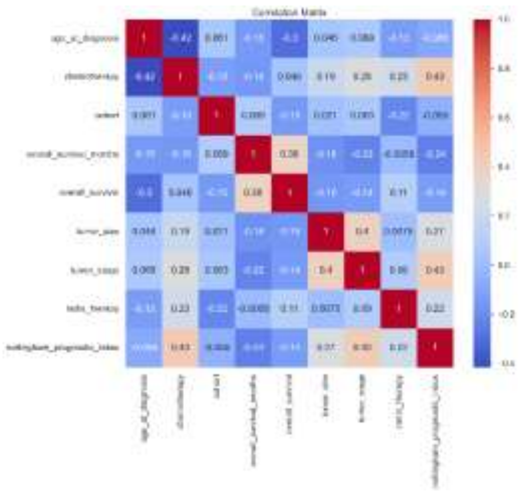
Keeping outliers in healthcare data is often crucial, as they can provide valuable insights into rare conditions, anomalies, or critical cases that may be clinically significant. However, we still standardize the data during training as Standardization is used to bring all attributes to a common scale without distorting differences in the ranges of values. This is especially important when dealing with clinical data, which may have attributes with vastly different scales.



Visualizing clinical data:



The largest group is those who are living, with a count close to 800.The second largest group is those who died of disease, with a count slightly above 600.The smallest group is those who died of other causes, with a count slightly above 400.A significant portion of the dataset consists of individuals who are still living. Among those who have died, more individuals died from the disease compared to other causes. This distribution can provide insights into the effectiveness of treatments or the severity of the disease.



Age at Diagnosis:

Negatively correlated with overall_survival (-0.46), indicating that older age at diagnosis is associated with lower overall survival.

Positively correlated with nottingham_prognostic_index (0.88), suggesting that older age at diagnosis is associated with a higher prognostic index.

Chemotherapy:

Negatively correlated with overall_survival (-0.13), indicating that patients who received chemotherapy tend to have lower overall survival.

Overall Survival Months:

Positively correlated with overall_survival (0.38), indicating that longer survival months are associated with overall survival.

Tumor Size:

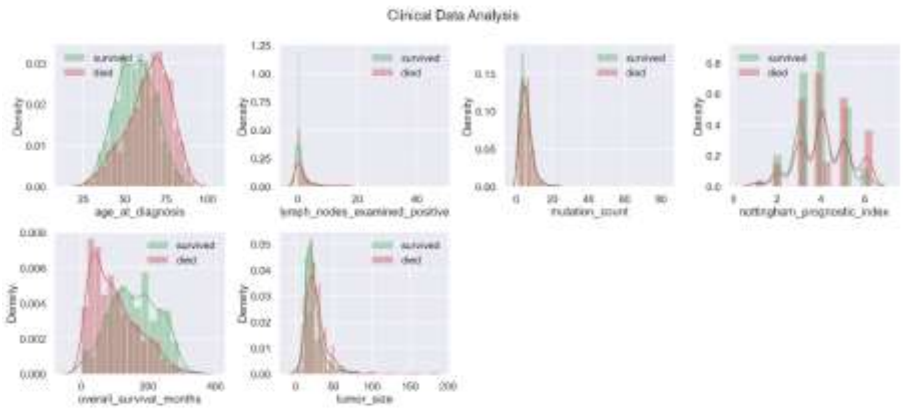
Positively correlated with tumor_stage (0.69), indicating that larger tumor size is associated with a higher tumor stage.

Nottingham Prognostic Index:

Negatively correlated with overall_survival (-0.55), indicating that a higher prognostic index is associated with lower overall survival.

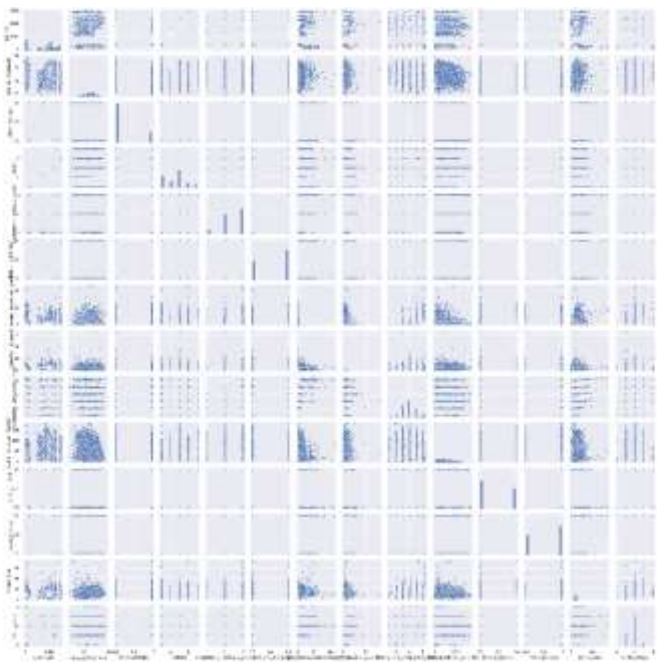
Insight:

The heatmap provides a visual representation of the relationships between different clinical variables. Strong correlations (both positive and negative) can help identify key factors that influence patient outcomes. For example, age at diagnosis and the Nottingham Prognostic Index are important factors related to overall survival.



- **Age at Diagnosis**: Younger age at diagnosis is associated with higher survival rates.
- **Lymph Nodes Examined Positive**: Fewer positive lymph nodes are associated with higher survival rates.
- **Mutation Count**: Higher mutation counts may be slightly associated with lower survival rates.
- **Nottingham Prognostic Index**: Lower values are associated with higher survival rates.
- **Overall Survival Months**: Longer survival months are associated with higher survival rates.
- **Tumor Size**: Smaller tumor sizes are associated with higher survival rates.

These distribution plots provide a clear visual representation of how different clinical variables are distributed among patients who survived and those who died. This can help in identifying key factors that influence patient outcomes and guide further analysis or treatment strategies.



- **Correlations**: The pair plot helps in identifying potential correlations between variables. For example, there is a noticeable positive correlation between age at diagnosis and the Nottingham Prognostic Index.
- **Distributions**: The diagonal elements provide insights into the distribution of each variable, helping to identify any skewness or outliers.
- **Relationships**: The scatter plots help in visualizing the relationships between pairs of variables, which can guide further statistical analysis or modeling efforts.

Data Preprocessing before training:

Before proceeding with the analysis, we performed preprocessing steps to clean and prepare the data. One of the initial steps involved dropping certain columns from the dataset that were deemed irrelevant or redundant for our analysis.

Handling Categorical Data:

We used a code segment that performs various data preprocessing tasks on a DataFrame named clinical_df. Initially, it identifies columns with categorical data by selecting those with the data type 'object' using the select_dtypes method and converts them into a list. Subsequently, it removes specific columns labeled as unwanted, such as 'patient_id' and 'death_from_cancer', from the list of categorical columns.

The next step involves creating dummy variables for each categorical column using one-hot encoding, except for the 'patient_id' column which is dropped. The resulting DataFrame is stored in dummies_clinical_df. The code then defines a function named combine_columns that helps in consolidating the dummy variables created into single columns based on their prefixes. This function is applied iteratively for each categorical column to generate new consolidated columns such as 'type_of_breast_surgery', 'cancer_type', 'cancer_type_detailed', and 'primary_tumor_laterality'.

Following this, certain original boolean columns are identified and dropped from the DataFrame, ensuring that only the consolidated columns remain. Subsequently, the categorical columns are converted into numerical format using the astype method along with the cat.codes attribute, which assigns unique numerical codes to each category within the column.

Finally, the modified DataFrame is printed to display the changes made. This series of preprocessing steps transforms categorical data into a format suitable for machine learning algorithms, facilitating further analysis and modeling tasks.

Basic Training vs. Hyperparameter Tuning:

Basic Training trains each model with its default hyperparameters.There is no hyperparameter tuning or cross-validation involved. It simply splits the data, trains the models, and evaluates them on the test set. Provides basic metrics such as accuracy, classification report, and confusion matrix.

Hyperparameter Tuning involves hyperparameter tuning using GridSearchCV for each model. It searches for the best hyperparameters based on cross-validation on the training set. Evaluates the best model found during the grid search on the test set. Provides detailed results including best parameters, cross-validation accuracy, test accuracy, classification report, and confusion matrix.

Data Preparation

Basic Training:

Uses a SimpleImputer to handle missing values and a StandardScaler to standardize the features. Both imputation and scaling are done before training and testing the models.

Hyperparameter Tuning:

The snippet does not explicitly show data preprocessing steps like imputation and scaling. thus, each model's training is wrapped in a Pipeline that includes steps for SimpleImputer and StandardScaler, followed by the model itself. This ensures that imputation and scaling are applied both during cross-validation and when evaluating the test set.

Model Evaluation and Reporting

Basic Training:

Directly trains and evaluates the models on the test set after the initial split.

Prints accuracy, classification report, and confusion matrix for each model.

Simpler and quicker to run, but might not give the best model performance due to the lack of hyperparameter tuning.

Hyperparameter Tuning:

Uses GridSearchCV to perform a comprehensive search over specified hyperparameter grids.

Evaluates the models using cross-validation, which provides a more robust measure of model performance.

Summarizes the results, including best parameters and both cross-validation and test accuracies.

Also generates a bar plot of test accuracies for a visual comparison.

Model Training Loop

Basic Training:

Iterates over the models dictionary, trains each model, and evaluates it immediately.

Hyperparameter Tuning:

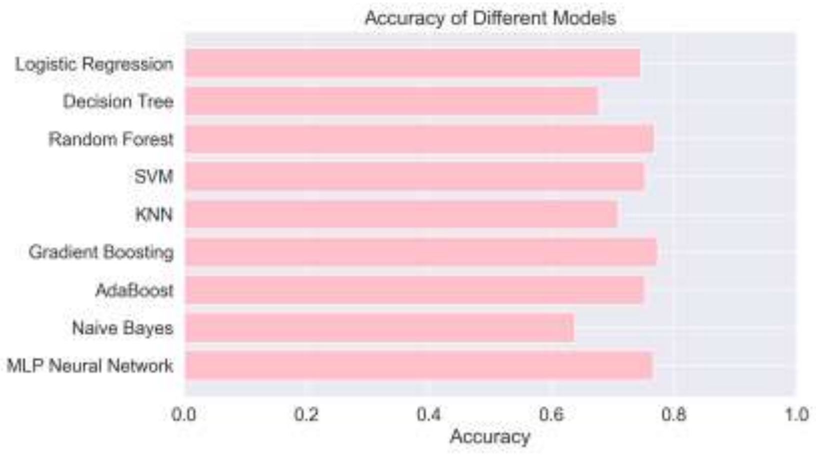
Iterates over the parameter grids, performs grid search for each model, finds the best model, and then evaluates it.

Code Comparison Summary

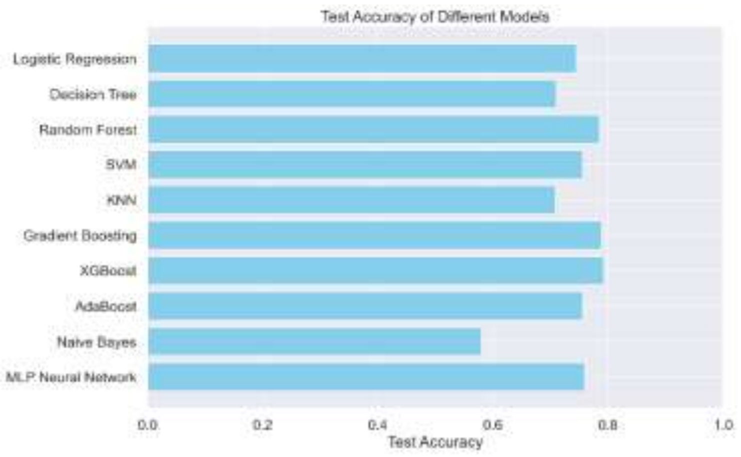
Basic Training is simpler and faster but lacks the depth of model evaluation and optimization. **Hyperparameter Tuning** is more comprehensive, involving hyperparameter tuning and cross-validation, leading to potentially better model performance and more reliable evaluation metrics.

Best model for each code

Basic Training: Based on the test accuracies, Random Forest has the highest accuracy of 0.77



Hyperparameter Tuning: Random Forest, Gradient Boosting, and XGBoost achieve the highest test accuracy among the models tested, each with a test accuracy of 0.79.

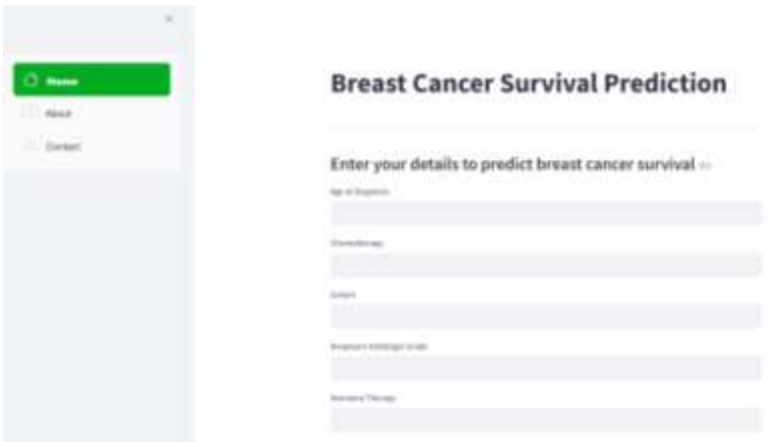


Thus, for our model deployment we decided to rely on: Random Forrest

Model Deployment

We wrote the Streamlit application code in a Python file, for instance, app.py and create a sidebar navigation menu for different pages.

Home Page for Prediction: Create the main page where users can input their details for prediction



Breast Cancer Survival Prediction

Enter your details to predict breast cancer survival

Age at Diagnosis

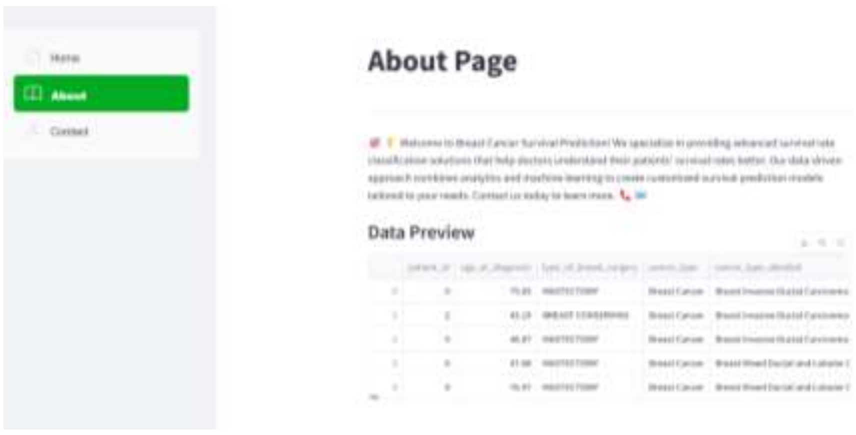
Chemotherapy

Cohort

Neoplasm Histologic Grade

Hormone Therapy

About and Contact Pages: Add additional pages to provide information about the project and contact options.

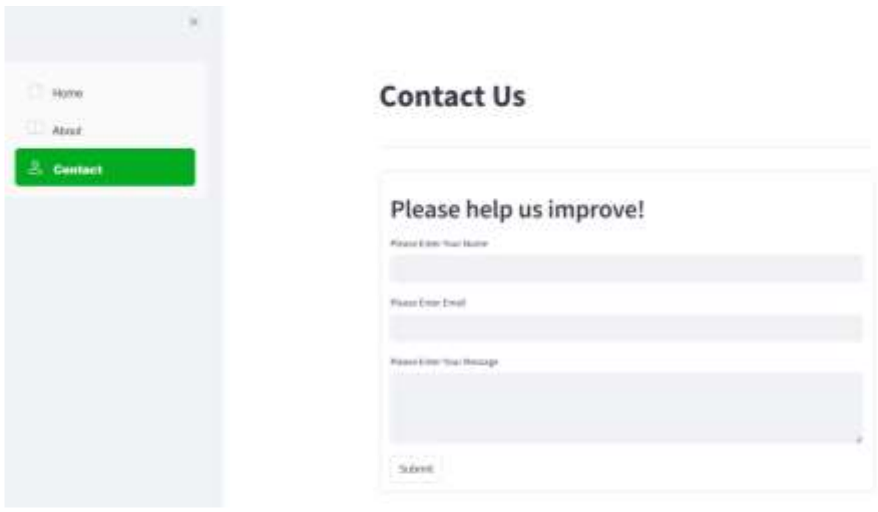


About Page

Welcome to Breast Cancer Survival Prediction! We specialize in providing advanced survival rate classification solutions that help doctors understand their patients' survival rates better. Our data-driven approach combines analytics and machine learning to create customized survival prediction models tailored to your needs. Contact us today to learn more. 📞 ✉️

Data Preview

	patient_id	age_at_diagnosis	type_of_breast_surgery	cancer_type	cancer_type_detailed
0	0	75.65	MASTECTOMY	Breast Cancer	Breast Invasive Ductal Carcinoma
1	2	43.19	BREAST CONSERVING	Breast Cancer	Breast Invasive Ductal Carcinoma
2	5	48.87	MASTECTOMY	Breast Cancer	Breast Invasive Ductal Carcinoma
3	6	47.68	MASTECTOMY	Breast Cancer	Breast Mixed Ductal and Lobular C
4	8	76.97	MASTECTOMY	Breast Cancer	Breast Mixed Ductal and Lobular C



Contact Us

Please help us improve!

Please Enter Your Name

Please Enter Email

Please Enter Your Message