2024

DECISION ANALYSIS

OPTIMIZING BREAST CANCER PREDICTION: BEST DATA MODELS FOR DIFFERENT DATASET TYPES

NATALIA LAIRAN, BENECIA JEAN, EUGENIA ANCILLA

REPORT

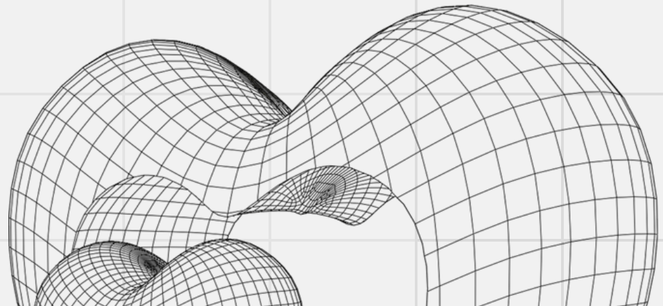


TABLE OF CONTENT

BACKGROUND

Introduction of Topic

- Breast cancer remains one of the most life-threatening cancers affecting women worldwide.
- Accurate and early diagnosis of breast cancer is essential for effective treatment and better patient outcomes.

Research Area

- Explore, compare, and identify the most suitable machine learning for different type of datasets in the context of breast cancer prediction.

DATASETS

Numerical

Breast Cancer Dataset

Binary Classification Prediction for type of Breast Cancer

Categorical

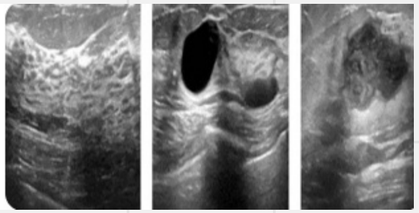
Breast Cancer Dataset

Breast Cancer Data for Classification Problems

Image

Breast Ultrasound Images Dataset

Breast ultrasound images for classification, detection & segmentation



BREAST CANCER WISCONSIN (DIAGNOSTIC) DATASET

id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_me	compactness_m	concavity_mean	concave points_	symmetry_mear	fractal_dimensic	radius_se	texture_se	perimeter_se	area_se
842302	M	17.99	10.38	122.8	1001	0.1184	0.2776	0.3001	0.1471	0.2419	0.07871	1.095	0.9053	8.589	153.4
842517	M	20.57	17.77	132.9	1326	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	0.5435	0.7339	3.398	74.08
84300903	M	19.69	21.25	130	1203	0.1096	0.1599	0.1974	0.1279	0.2069	0.05999	0.7456	0.7869	4.585	94.03
84348301	M	11.42	20.38	77.58	386.1	0.1425	0.2839	0.2414	0.1052	0.2597	0.09744	0.4956	1.156	3.445	27.23
84358402	M	20.29	14.34	135.1	1297	0.1003	0.1328	0.198	0.1043	0.1809	0.05883	0.7572	0.7813	5.438	94.44
843786	M	12.45	15.7	82.57	477.1	0.1278	0.17	0.1578	0.08089	0.2087	0.07613	0.3345	0.8902	2.217	27.19

smoothness_se	compactness_s	concavity_se	concave points_	symmetry_se	fractal_dimensic	radius_worst	texture_worst	perimeter_worst	area_worst	smoothness_wo	compactness_w	concavity_worst	concave points_	symmetry_wors	fractal_dimension_wo
0.006399	0.04904	0.05373	0.01587	0.03003	0.006193	25.38	17.33	184.6	2019	0.1622	0.6656	0.7119	0.2654	0.4601	0.1189
0.005225	0.01308	0.0186	0.0134	0.01389	0.003532	24.99	23.41	158.8	1956	0.1238	0.1866	0.2416	0.186	0.275	0.08902
0.00615	0.04006	0.03832	0.02058	0.0225	0.004571	23.57	25.53	152.5	1709	0.1444	0.4245	0.4504	0.243	0.3613	0.08758
0.00911	0.07458	0.05661	0.01867	0.05963	0.009208	14.91	26.5	98.87	567.7	0.2098	0.8663	0.6869	0.2575	0.6638	0.173
0.01149	0.02461	0.05688	0.01885	0.01756	0.005115	22.54	16.67	152.2	1575	0.1374	0.205	0.4	0.1625	0.2364	0.07678

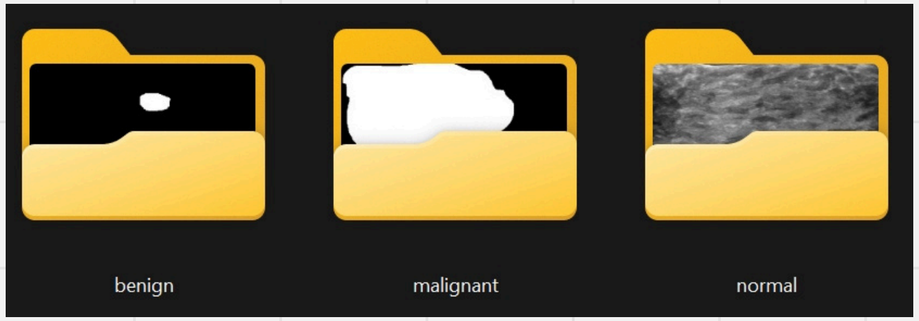
HTTPS://WWW.KAGGLE.COM/DATASETS/YASSERH/BREAST-CANCER-DATASET

BREAST CANCER DATASET OBTAINED FROM THE UNIVERSITY MEDICAL CENTRE

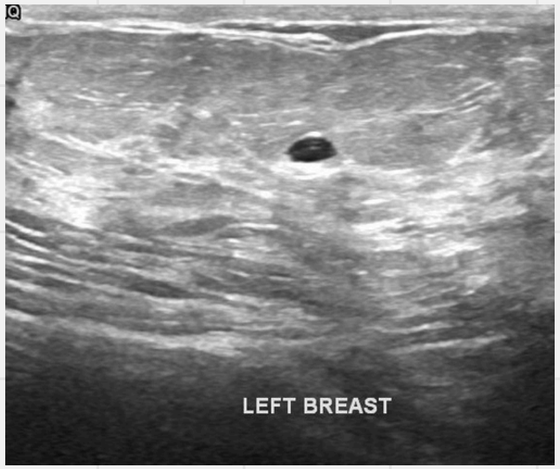
age	menopause	tumer-size	inv-nodes	node-caps	deg-malig	breast	breast-quad	irradiate	class
40-49'	premeno'	15-19'	0-2'	yes'	3'	right'	left_up'	no'	recurrence-eve
50-59'	ge40'	15-19'	0-2'	no'	1'	right'	central'	no'	no-recurrence-
50-59'	ge40'	35-39'	0-2'	no'	2'	left'	left_low'	no'	recurrence-eve

HTTPS://WWW.KAGGLE.COM/DATASETS/FAYSALMIAH1721758/BREAST-CANCER-DATA

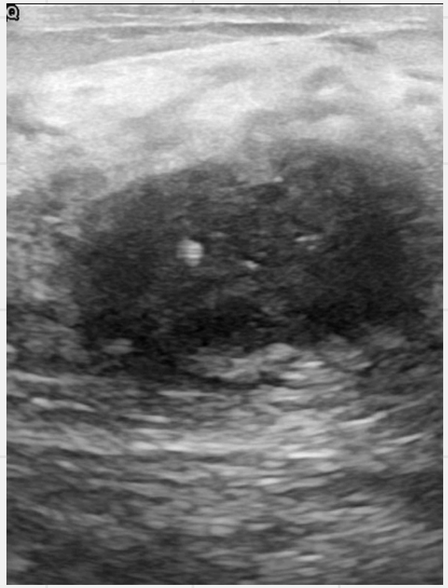
BREAST ULTRASOUND IMAGES



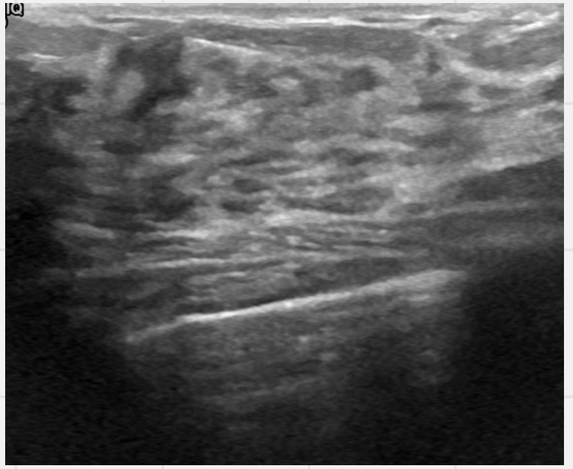
BENIGN



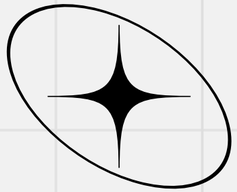
MALIGNANT



NORMAL



HTTPS://WWW.KAGGLE.COM/DATASETS/FAYSALMIAH1721758/BREAST-CANCER-DATA



DATA PREPROCESSING

Machine Learning & Deep Learning

PIPELINE:

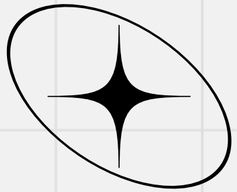
```python
def data_transformation(data):
    data = data.dropna()

    # Handle diagnosis and class mappings
    if "diagnosis" in data.columns:
        diagnosis_mapping = {"B": 0, "M": 1}
        data["diagnosis"] = data["diagnosis"].map(diagnosis_mapping)
        if "id" in data.columns:
            data = data.drop("id", axis=1)

    if "class" in data.columns:
        class_mapping = {"no-recurrence-events'": 0, "recurrence-events'": 1}
        data["class"] = data["class"].map(class_mapping)

    labels = None
    if "diagnosis" in data.columns:
        labels = data["diagnosis"]
        data = data.drop("diagnosis", axis=1)
    elif "class" in data.columns:
        labels = data["class"]
        data = data.drop("class", axis=1)

    features = list(data.columns)

    # Missing data imputation using median
    from sklearn.impute import SimpleImputer

    df_num = data.select_dtypes(include=[np.number])

    # Debugging: Print df_num information
    print("Numerical Features:")
    print(df_num.head())
    print("Numerical Features Shape:", df_num.shape)
    print("Numerical Features Dtypes:", df_num.dtypes)
```

PIPELINE:

```
# Debugging: Print df_num information
print("Numerical Features:")
print(df_num.head())
print("Numerical Features Shape:", df_num.shape)
print("Numerical Features Dtypes:", df_num.dtypes)

if not df_num.empty:
    imputer = SimpleImputer(strategy="median")
    df_num_imp = imputer.fit_transform(df_num)

    # Scaling using StandardScaler
    from sklearn.preprocessing import StandardScaler
    scaler = StandardScaler()
    df_num_sc = scaler.fit_transform(df_num_imp)

    # Debugging: Print shapes after transformation
    print("Shape after imputation and scaling:", df_num_sc.shape)

else:
    df_num_sc = None
    print("No numerical features found after filtering.")
```

PIPELINE:

```
# Encoding categorical features using OneHotEncoder
if "age" in data.columns:
    category_orders = {
        'age': ['20-29', '30-39', '40-49', '50-59', '60-69', '70-79'],
        'tumer-size': ['0-4', '5-9', '10-14', '15-19', '20-24', '25-29', '30-34', '35-39', '40-44', '45-49', '50-54'],
        'inv-nodes': ['0-2', '3-5', '6-8', '9-11', '12-14', '15-17', '18-20', '21-23', '24-26'],
        'deg-malig': [1, 2, 3]
    }
    category_mappings = {key: {label: idx for idx, label in enumerate(values)} for key, values in category_orders.items()}

    for column, mapping in category_mappings.items():
        data[column] = data[column].map(mapping)

    unordered_attributes = ['menopause', 'node-caps', 'breast', 'breast-quad', 'irradiate']
    df_unorder = data[unordered_attributes]

    from sklearn.preprocessing import OneHotEncoder
    encoder = OneHotEncoder(handle_unknown='ignore')
    df_unord_enc = encoder.fit_transform(df_unorder)

    # Print number of features after encoding
    print("Number of features after encoding:", df_unord_enc.shape[1])
    new_features = encoder.get_feature_names_out(unordered_attributes)
    features += new_features.tolist()

    # Remove categorical columns from features
    for feature in ["age", "menopause", "tumer-size", "inv-nodes", "node-caps", "deg-malig", "breast", "breast-quad", "irradiate"]:
        if feature in features:
            features.remove(feature)

    # Concatenate numerical and encoded categorical data
    if df_num_sc is not None:
        output = np.hstack([df_num_sc, df_unord_enc.toarray()])  # Convert sparse matrix to dense array
    else:
        output = df_unord_enc.toarray()  # Convert sparse matrix to dense array if necessary

else:
    # Only numerical data path
    if df_num_sc is not None:
        output = df_num_sc
    else:
        output = None  # Handle this case based on your requirement

return output, labels, features
```



MODELS

Machine Learning & Deep Learning

SUPPORT VECTOR CLASSIFIER

- SVC is a type of Support Vector Machine (SVM) used for classification tasks. It works by finding a hyperplane that best separates the classes in the feature space. It can handle linear and non-linear classification through the use of kernel functions.
- Best for: Binary classification with clear class separation and small-to-medium-sized datasets.

LOGISTIC REGRESSION

- Logistic regression is a statistical model used for binary classification. It models the probability that a given input belongs to a particular class using a logistic function.
- Best for: Binary classification problems like spam detection and disease prediction.

NAIVE BAYES

- Naive Bayes is a probabilistic classifier based on Bayes' theorem, with the assumption of independence between the features. It’s called "naive" because it assumes that the features are independent given the class label.
- Best for: Text classification tasks such as spam filtering and sentiment analysis.

DECISION TREE

- A decision tree is a flowchart-like structure where each internal node represents a test on a feature, each branch represents an outcome of the test, and each leaf node represents a class label. It’s easy to interpret and can handle both numerical and categorical data.
- Best for: Tasks requiring interpretability, like credit scoring and medical diagnosis.

RANDOM FOREST

- Random Forest is an ensemble learning method that constructs multiple decision trees during training and outputs the mode of the classes (classification) or mean prediction (regression) of the individual trees. It reduces overfitting and improves accuracy.
- Best for: High-dimensional data and tasks where robustness and accuracy are important, such as image classification and finance.

MULTI-LAYER PERCEPTRON

- MLP is a type of feedforward artificial neural network with one or more layers of neurons (nodes). Each node in one layer connects with a certain weight to every node in the following layer. It’s used for both classification and regression tasks.
- Best for: Tasks requiring complex feature transformations, like speech recognition and non-linear regression problems.

K-NEAREST NEIGHBOUR

- KNN is a simple, instance-based learning algorithm where the input consists of the k closest training examples in the feature space. The output is a class label for classification or a value for regression based on the majority class or average of k neighbors.
- Best for: Small datasets with clear class separation, such as pattern recognition and recommendation systems.

ADABOOST

- AdaBoost is an ensemble learning technique that combines multiple weak classifiers to create a strong classifier. It focuses on the misclassified samples by adjusting the weights of incorrectly classified instances, so future classifiers focus more on those cases.
- Best for: Boosting the performance of weak classifiers, particularly in binary classification tasks like fraud detection.

GRADIENT BOOSTING

- Gradient Boosting is another ensemble technique that builds models sequentially. Each new model corrects the errors of the previous models by fitting to the residual errors. It’s used for both classification and regression.
- Best for: Both classification and regression tasks where high accuracy is required, such as anomaly detection and web search ranking.

HISTOGRAM GRADIENT BOOSTING

- This is a variant of gradient boosting that uses histograms to speed up the training process by discretizing continuous features into bins. It’s efficient for large datasets and helps in managing memory usage.
- Best for: Large datasets with continuous features, offering efficient and scalable training.

BAGGING

- Bagging is an ensemble method that creates multiple versions of a predictor by sampling the training data with replacement (bootstrap sampling) and combining their results (usually by averaging or voting). It reduces variance and helps in avoiding overfitting.
- Best for: Reducing variance and improving stability in models like decision trees, useful in finance and bioinformatics.

EXTRA TREE

- Extra Trees is similar to random forest but differs in the way splits are chosen. Instead of searching for the best split among a random subset of features, splits are chosen completely at random. This can lead to greater variance reduction at the cost of some increase in bias.
- Best for: High-dimensional and large datasets where variance reduction is crucial, such as in genomics and high-frequency trading.

VOTING

- Voting is an ensemble method where multiple models (classifiers) are trained and their predictions are combined. In majority voting (classification), the class that gets the most votes is chosen. For regression, the average of the predictions is taken.
- Best for: Improving overall model performance by combining the strengths of different classifiers, used in general-purpose classification tasks.

STACKING

- Stacking (Stacked Generalization) is an ensemble method where multiple models are trained and a meta-learner is used to combine their predictions. The base models' outputs are fed as input features to the meta-learner, which makes the final prediction.
- Best for: Tasks where combining predictions from diverse models can improve performance, like complex classification problems and competitions.

RESULTS

Numerical Dataset														
	SVC	Logistic Regression	Naive Bayes	Decision Tree	Random Forest	MLP	KNN	ADABoost	Gradient Boosting	Histogram Gradient Boosting	Bagging	Extra Tree	Stacking	Voting
Accuracy	0.9734513274	0.982300885	0.9469026549	0.9380530973	0.9557522124	0.982300885	0.982300885	0.9734513274	0.9734513274	0.9911504425	0.9646017699	0.982300885	0.9646017699	0.982300885
Precision	0.9523809524	0.9756097561	0.9069767442	0.8863636364	0.9090909091	0.9756097561	0.9756097561	0.9523809524	0.9523809524	1	0.9302325581	0.9756097561	0.9302325581	0.9756097561
Recall	0.9756097561	0.9756097561	0.9512195122	0.9512195122	0.9756097561	0.9756097561	0.9756097561	0.9756097561	0.9756097561	0.9756097561	0.9756097561	0.9756097561	0.9756097561	0.9756097561
F1 Score	0.9638554217	0.9756097561	0.9285714286	0.9176470588	0.9411764706	0.9756097561	0.9756097561	0.9638554217	0.9638554217	0.987654321	0.9523809524	0.9756097561	0.9523809524	0.9756097561
AUC	0.9972899729	0.9959349593	0.9788279133	0.9408875339	0.9949186992	0.9969512195	0.9993224932	0.9918699187	0.9949186992	0.9966124661	0.9920392954	0.9959349593	0.9940718157	0.9942411924
Log-loss	0.06053120028	0.05703160022	0.9209823852	2.232792688	0.1014855054	0.05711115776	0.02662083888	0.5010486368	0.1576272419	0.09656883063	0.1060483393	0.08411060695	0.07425440024	0.1717546611
Confusion Matri:	[[70 2] [1 40]]	[[71 1] [1 40]]	[[68 4] [2 39]]	[[67 5] [2 39]]	[[68 4] [1 40]]	[[71 1] [1 40]]	[[71 1] [1 40]]	[[70 2] [1 40]]	[[70 2] [1 40]]	[[72 0] [1 40]]	[[69 3] [1 40]]	[[71 1] [1 40]]	[[69 3] [1 40]]	[[71 1] [1 40]]

The best model for numerical dataset is Histogram Gradient Boosting. It is the best model, as it achieves the highest accuracy (0.9912), AUC (0.9968), and consistently high scores across other metrics (precision, recall, F1 score), indicating it effectively captures the complex relationships and interactions within the numerical data.

RESULTS

Categorical Dataset														
	SVC	Logistic Regression	Naive Bayes	Decision Tree	Random Forest	MLP	KNN	ADABoost	Gradient Boosting	Histogram Gradient Boosting	Bagging	Extra Tree	Stacking	Voting
Accuracy	0.6481481481	0.6481481481	0.6666666667	0.6481481481	0.6666666667	0.7407407407	0.6481481481	0.6481481481	0.6666666667	0.7407407407	0.6666666667	0.6481481481	0.7222222222	0.7222222222
Precision	0.375	0.3333333333	0.4615384615	0.375	0.4444444444	0.6153846154	0.375	0.3333333333	0.4444444444	0.6153846154	0.4	0.375	0.625	0.625
Recall	0.1764705882	0.1176470588	0.3529411765	0.1764705882	0.2352941176	0.4705882353	0.1764705882	0.1176470588	0.2352941176	0.4705882353	0.1176470588	0.1764705882	0.2941176471	0.2941176471
F1 Score	0.24	0.1739130435	0.4	0.24	0.3076923077	0.5333333333	0.24	0.1739130435	0.3076923077	0.5333333333	0.1818181818	0.24	0.4	0.4
AUC	0.6081081081	0.5699523052	0.6128775835	0.6478537361	0.6812400636	0.6780604134	0.6383147854	0.613672496	0.6526232114	0.6383147854	0.6414944356	0.6478537361	0.6128775835	0.6669316375
Log-loss	0.6526507335	0.6173297222	1.165371505	4.438053636	0.7484761774	0.6427290519	4.438706764	0.6201422754	1.373759148	0.8542659604	0.6191725437	4.438531983	0.6382883665	0.5953356428
Confusion Matri:	[[32 5] [14 3]]	[[33 4] [15 2]]	[[30 7] [11 6]]	[[32 5] [14 3]]	[[32 5] [13 4]]	[[32 5] [9 8]]	[[32 5] [14 3]]	[[33 4] [15 2]]	[[32 5] [13 4]]	[[32 5] [9 8]]	[[34 3] [15 2]]	[[32 5] [14 3]]	[[34 3] [12 5]]	[[34 3] [12 5]]

The best model for categorical dataset is Historam Gradient Boosting. It is also the best model for the categorical dataset, with a strong AUC (0.8543) and balanced performance across accuracy, precision, recall, and F1 score. Its ability to handle encoded categorical data and capture complex patterns makes it the most effective.

RESULTS

Image Dataset														
	SVC	Logistic Regression	Naive Bayes	Decision Tree	Random Forest	MLP	KNN	ADABoost	Gradient Boosting	Histogram Gradient Boosting	Bagging	Extra Tree	Stacking	Voting
Accuracy	0.7594936709	0.4810126582	0.4050632911	0.9380530973	0.7594936709	0.582278481	0.5316455696	0.7848101266	0.7088607595	0.5696202532	0.582278481	0.7594936709	0.7215189873	0.582278481
Precision	0.6069013352	0.5928816083	0.5748224761	0.9403004652	0.6069013352	0.6203693512	0.6361655773	0.7752377354	0.6500904159	0.6131117579	0.6228960913	0.6069013352	0.6793248945	0.5794269022
Recall	0.7594936709	0.4810126582	0.4050632911	0.9380530973	0.7594936709	0.582278481	0.5316455696	0.7848101266	0.7088607595	0.5696202532	0.582278481	0.7594936709	0.7215189873	0.582278481
F1 Score	0.6680810468	0.5027213635	0.4255437871	0.9384872684	0.6680810468	0.592798176	0.548464194	0.749157082	0.6679181864	0.5886215051	0.5752308942	0.6680810468	0.6915287244	0.5754651844
Confusion Matrix	[[54 0 0] [17 0 0] [2 0 6]]	[[24 12 18] [8 8 1] [2 0 6]]	[[22 6 26] [8 2 7] [0 0 8]]	[[36 16 2] [4 8 5] [1 1 6]]	[[54 0 0] [17 0 0] [2 0 6]]	[[35 12 7] [10 5 2] [1 1 6]]	[[26 16 12] [7 10 0] [1 1 6]]	[[52 2 0] [13 4 0] [2 0 6]]	[[48 4 2] [15 2 0] [1 1 6]]	[[36 18 0] [14 3 0] [1 1 6]]	[[37 3 14] [13 3 1] [2 0 6]]	[[54 0 0] [17 0 0] [2 0 6]]	[[48 5 1] [14 3 0] [1 1 6]]	[[37 9 8] [14 3 0] [1 1 6]]

The best model for image dataset is Decision Tree. It shows the highest accuracy (0.9381), precision (0.9430), recall (0.9381), and F1 score (0.9385). This suggests that it effectively classifies the image data, likely due to the specific feature extraction or characteristics of the dataset being well-suited to decision trees.

REFERENCES

https://www.kaggle.com/datasets/yasserh/breast-cancer-dataset/data
https://www.kaggle.com/datasets/faysalmiah1721758/breast-cancer-data
https://www.kaggle.com/datasets/aryashah2k/breast-ultrasound-images-dataset

LINK TO OUR FINAL CODE AND DATASET

https://binusianorg-my.sharepoint.com/personal/benecia_saputra_binus_ac_id/_layouts/15/guestaccess.aspx?share=ET9aj4u5b4xBi7ZVcE94phYBSDWq5HyB7Kv16PkGHpN7HA&e=VczM7I

THANK YOU