



ARTIFICIAL INTELLIGENCE ANALYSIS PROJECT USING BREAST CANCER WISCONSIN DATASET

Beyza Nur Yıldırım

INTRODUCTION

DATA EXPLORATION AND ANALYSIS (EDA)

Dataset Overview

The dataset used in this project is the **Breast Cancer Wisconsin (Diagnostic) Dataset** [1]. It consists of 569 observations with 32 features, including an id column, a diagnosis label indicating whether the tumor is malignant (M) or benign (B), and 30 real-valued features describing various tumor characteristics derived from digitized images of fine needle aspirate (FNA) of breast masses.

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.24
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.18
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.20
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.25
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.18

The dataset includes:

- **Mean features:** (e.g., **texture_mean**, **area_mean**) – representing average measurements.
- **Standard error features (*_se):** Indicating variation in measurement.
- **Worst-case features (*_worst):** Largest values recorded across tumor cells.

Structure and Cleaning

- The dataset initially contained an extra column **Unnamed: 32**, which was identified as redundant and removed.
- The id column was also dropped, as it contains no predictive value.
- The target column diagnosis was encoded as a binary variable: **Malignant (M) = 1**, **Benign (B) = 0**.

There were **no missing values** in the dataset, ensuring completeness for further analysis.

Summary Statistics

	count	mean	std	min	25%	50%	75%	max	skewness	kurtosis
id	569.0	3.037183e+07	1.250206e+08	8670.000000	869218.000000	906024.000000	8.813129e+06	9.113205e+08	6.473752	42.193194
diagnosis	569.0	3.725835e-01	4.839180e-01	0.000000	0.000000	0.000000	1.000000e+00	1.000000e+00	0.528461	-1.726811
radius_mean	569.0	1.412729e+01	3.524049e+00	6.981000	11.700000	13.370000	1.578000e+01	2.811000e+01	0.942380	0.845522
texture_mean	569.0	1.928965e+01	4.301036e+00	9.710000	16.170000	18.840000	2.180000e+01	3.928000e+01	0.650450	0.758319
perimeter_mean	569.0	9.196903e+01	2.429898e+01	43.790000	75.170000	86.240000	1.041000e+02	1.885000e+02	0.990650	0.972214
area_mean	569.0	6.548891e+02	3.519141e+02	143.500000	420.300000	551.100000	7.827000e+02	2.501000e+03	1.645732	3.652303
smoothness_mean	569.0	9.636028e-02	1.406413e-02	0.052630	0.086370	0.095870	1.053000e-01	1.634000e-01	0.456324	0.855975
compactness_mean	569.0	1.043410e-01	5.281276e-02	0.019380	0.064920	0.092630	1.304000e-01	3.454000e-01	1.190123	1.650130
concavity_mean	569.0	8.879932e-02	7.971981e-02	0.000000	0.029560	0.061540	1.307000e-01	4.268000e-01	1.401180	1.998638
concave points_mean	569.0	4.891915e-02	3.880284e-02	0.000000	0.020310	0.033500	7.400000e-02	2.012000e-01	1.171180	1.066556
symmetry_mean	569.0	1.811619e-01	2.741428e-02	0.106000	0.161900	0.179200	1.957000e-01	3.040000e-01	0.725609	1.287933
fractal_dimension_mean	569.0	6.279761e-02	7.060363e-03	0.049960	0.057700	0.061540	6.612000e-02	9.744000e-02	1.304489	3.005892
radius_se	569.0	4.051721e-01	2.773127e-01	0.111500	0.232400	0.324200	4.789000e-01	2.873000e+00	3.088612	17.686726
texture_se	569.0	1.216853e+00	5.516484e-01	0.360200	0.833900	1.108000	1.474000e+00	4.885000e+00	1.646444	5.349169
perimeter_se	569.0	2.866059e+00	2.021855e+00	0.757000	1.606000	2.287000	3.357000e+00	2.198000e+01	3.443615	21.401905
area_se	569.0	4.033708e+01	4.549101e+01	6.802000	17.850000	24.530000	4.519000e+01	5.422000e+02	5.447186	49.209077
smoothness_se	569.0	7.040979e-03	3.002518e-03	0.001713	0.005169	0.006380	8.146000e-03	3.113000e-02	2.314450	10.469840

Descriptive statistics were computed for all numeric variables. Key findings:

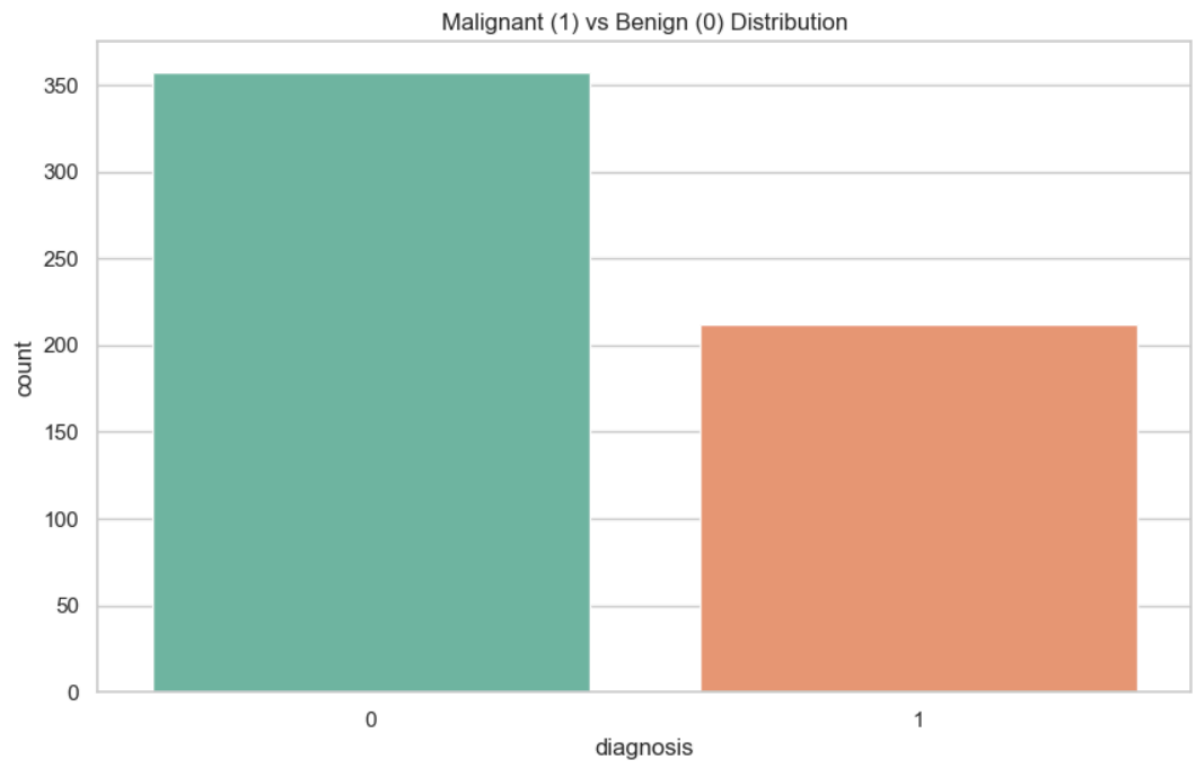
- Wide value ranges for several features:
 - **area_mean**: 143.5 – 2501.0
 - **perimeter_mean**: 43.79 – 188.5
 - **radius_mean**: 6.98 – 28.11
- Narrower ranges in features like **smoothness_mean**, **compactness_mean**, and **concavity_mean** (typically 0–0.4)
- Some features, such as **concavity_mean** and **concave points_mean**, had 0 values, indicating that some tumors exhibit no concavity.

Class Distribution

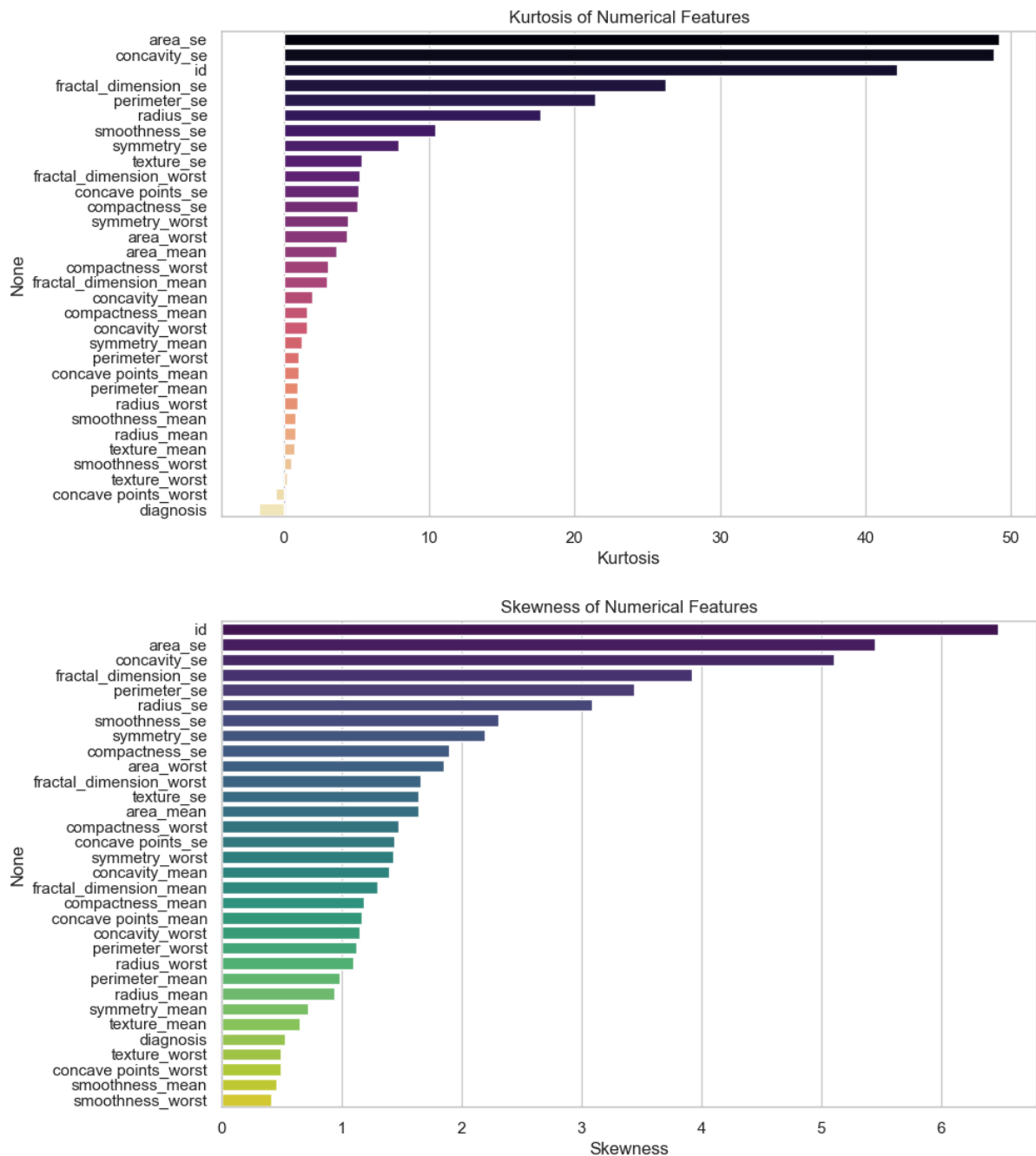
The dataset is **imbalanced**, with **62.7% benign** and **37.3% malignant** samples. This class imbalance was visualized using a count plot.

```
# class distribution
df['diagnosis'].value_counts(normalize=True)
```

```
diagnosis
B      0.627417
M      0.372583
Name: proportion, dtype: float64
```



Statistical Distributions



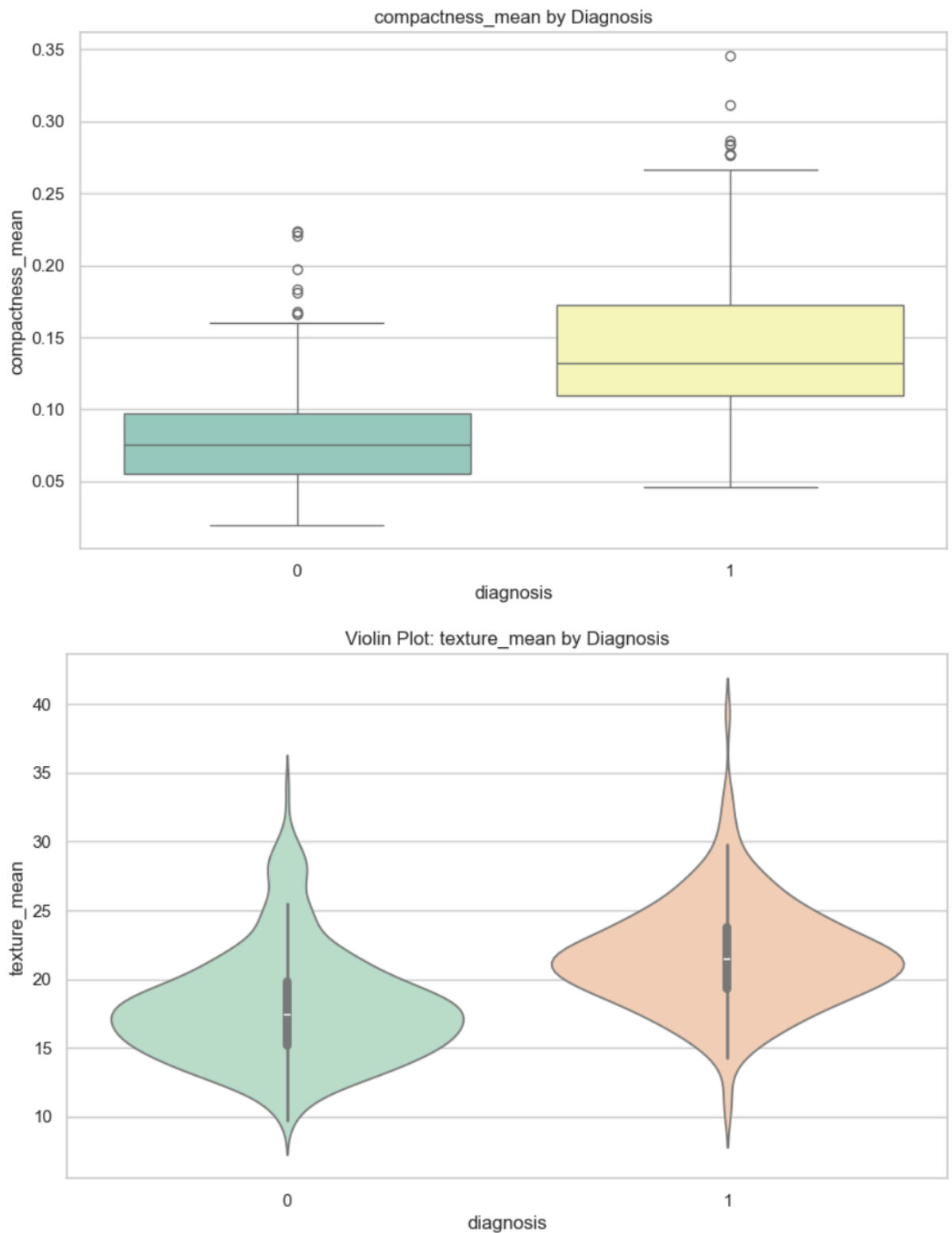
For all features, **skewness and kurtosis** values were calculated:

- **Positive skewness** was observed across most features, indicating the presence of more extreme high values.
- Features like **area_mean** had **high kurtosis**, suggesting heavy tails and **potential outliers**.

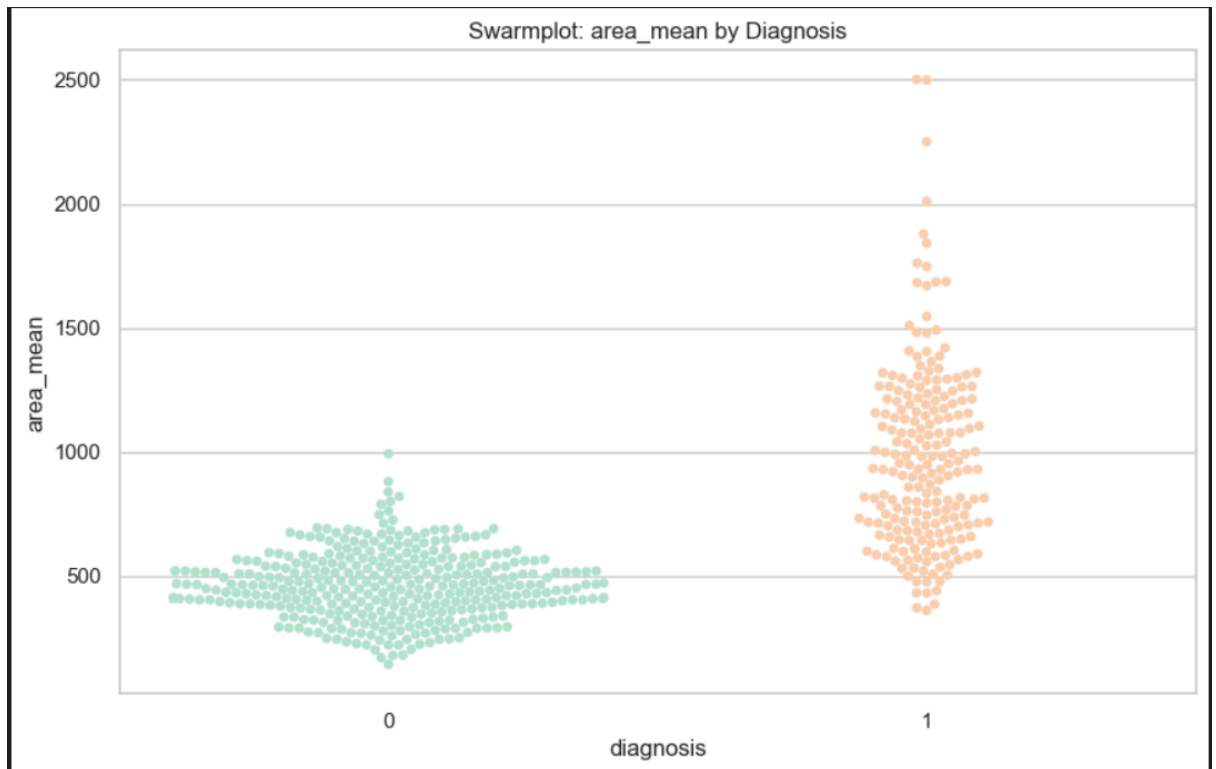
Visualizations

Several plots were generated to analyze the feature distributions and relationships:

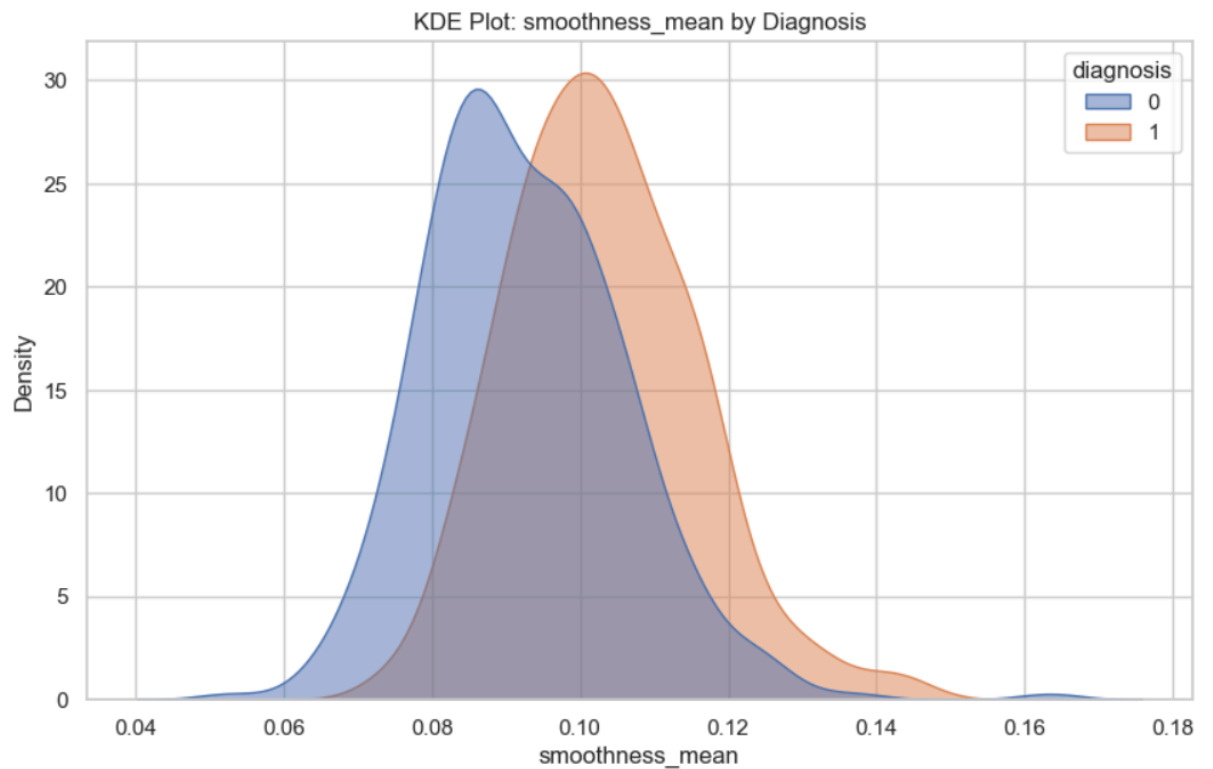
- **Boxplots and Violin plots:** Visualized distribution, central tendency, and spread of features across classes.



- **Swarmplots:** Illustrated clustering and outliers of tumor characteristics for each class.

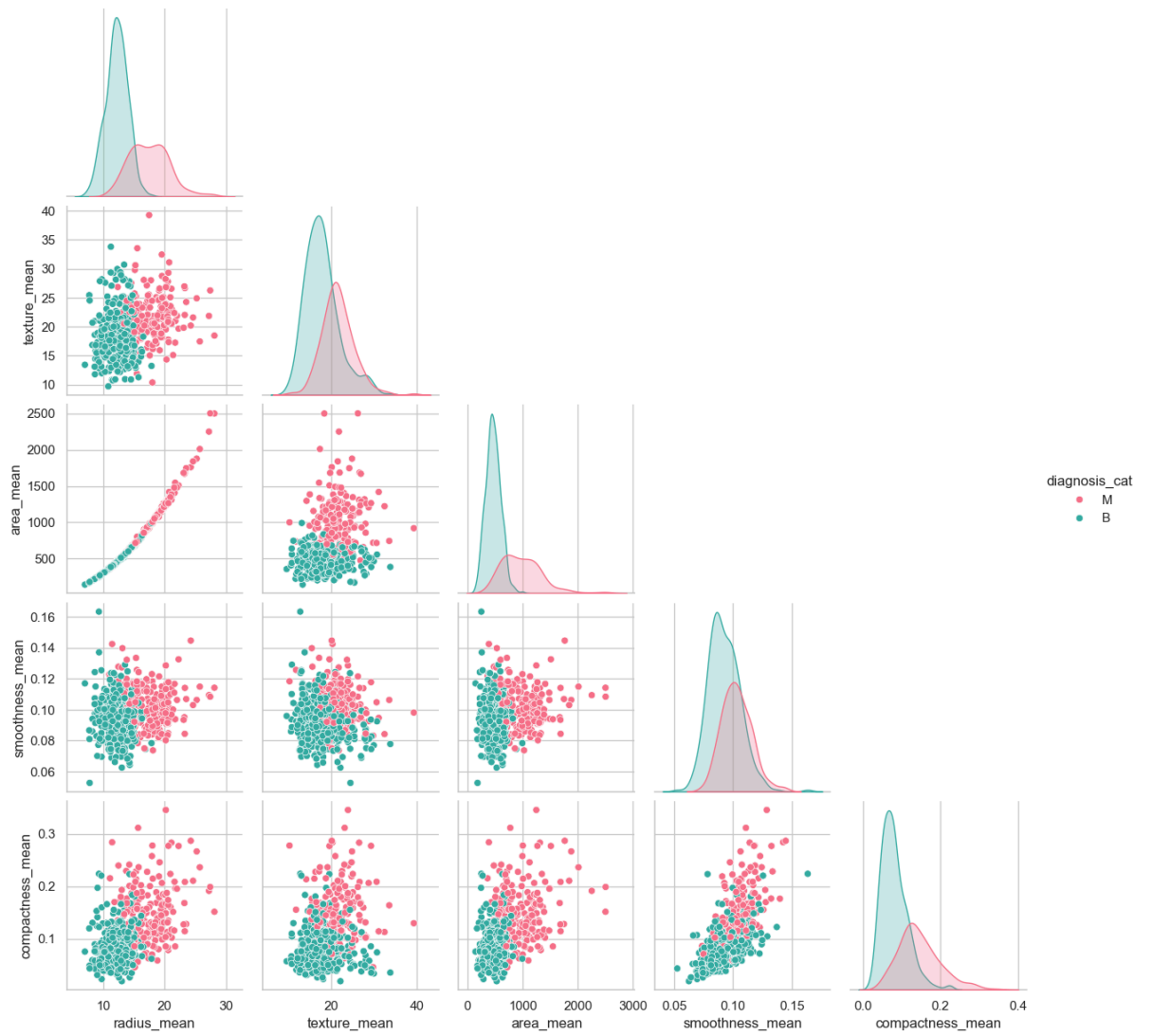


- **KDE plots:** Revealed the shape and overlap of feature distributions.

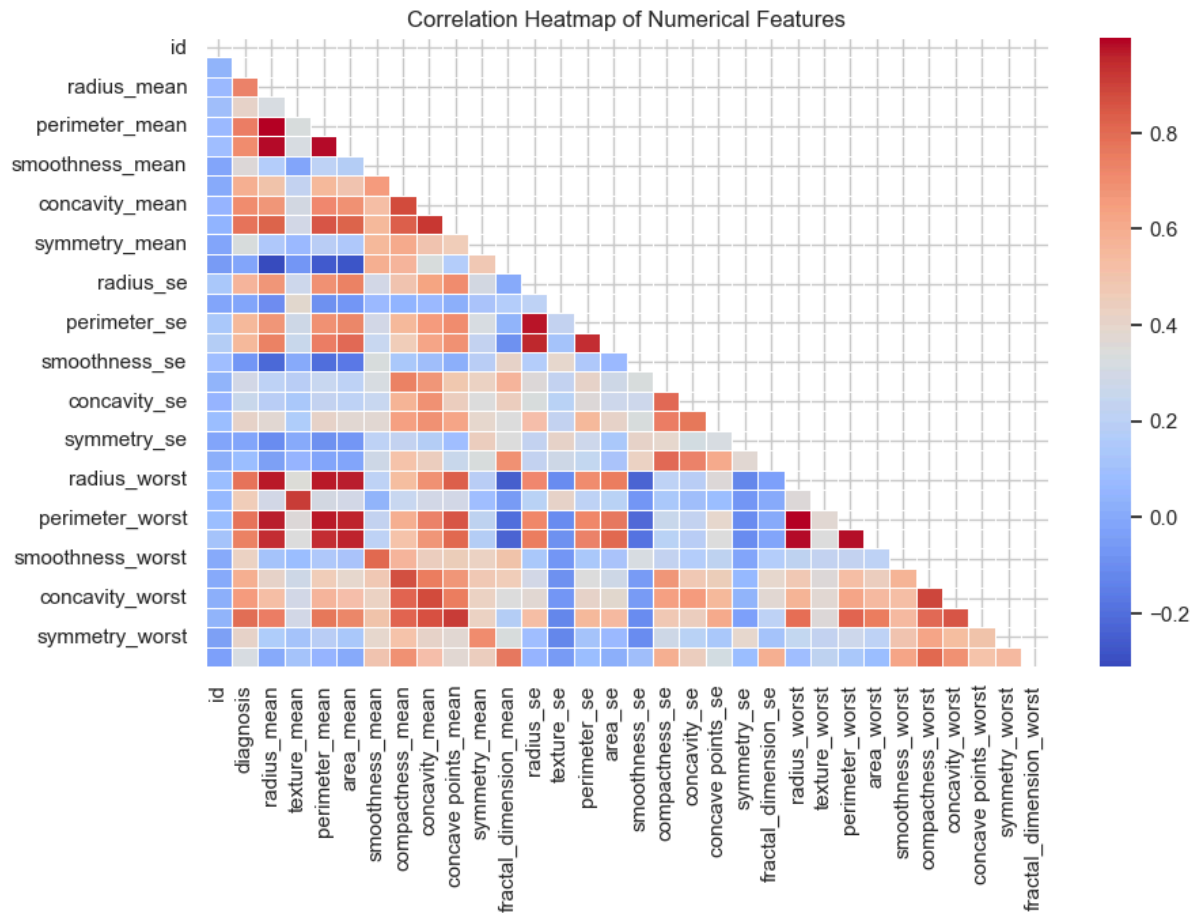


- **Pairplot:** Provided a multivariate view of the relationships between key features.

Pairplot of Selected Features by Diagnosis



- **Heatmap of correlations:** Showed that:



- **radius_mean, perimeter_mean, and area_mean** were strongly positively correlated.
- Similar strong correlations existed among ***_worst** features.
- Features like **smoothness_mean, symmetry_mean, and fractal_dimension_mean** showed low correlation with others.

DATA PREPROCESSING/ FEATURE ENGINEERING

To ensure the dataset was clean, well-behaved, and ready for training robust machine learning models, several preprocessing and feature engineering steps were applied to the Breast Cancer Wisconsin dataset. These steps include handling irrelevant features, encoding the target variable, outlier treatment, skewness correction, and scaling.

Handling Irrelevant and Redundant Features

Upon loading the dataset, the following columns were dropped:

- **id:** A unique identifier for each sample, which holds no predictive value.
- **Unnamed: 32:** An entirely null column that was deemed unnecessary for modeling.

These removals helped simplify the dataset by eliminating noise and non-informative features.

Encoding the Target Variable

```
# encode target
df["diagnosis"] = df["diagnosis"].map({"M": 1, "B": 0})
```

df

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	...
0	1	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419	...
1	1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812	...
2	1	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069	...
3	1	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597	...
4	1	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809	...
...
564	1	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	...
565	1	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	...
566	1	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	...
567	1	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	...
568	0	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587	...

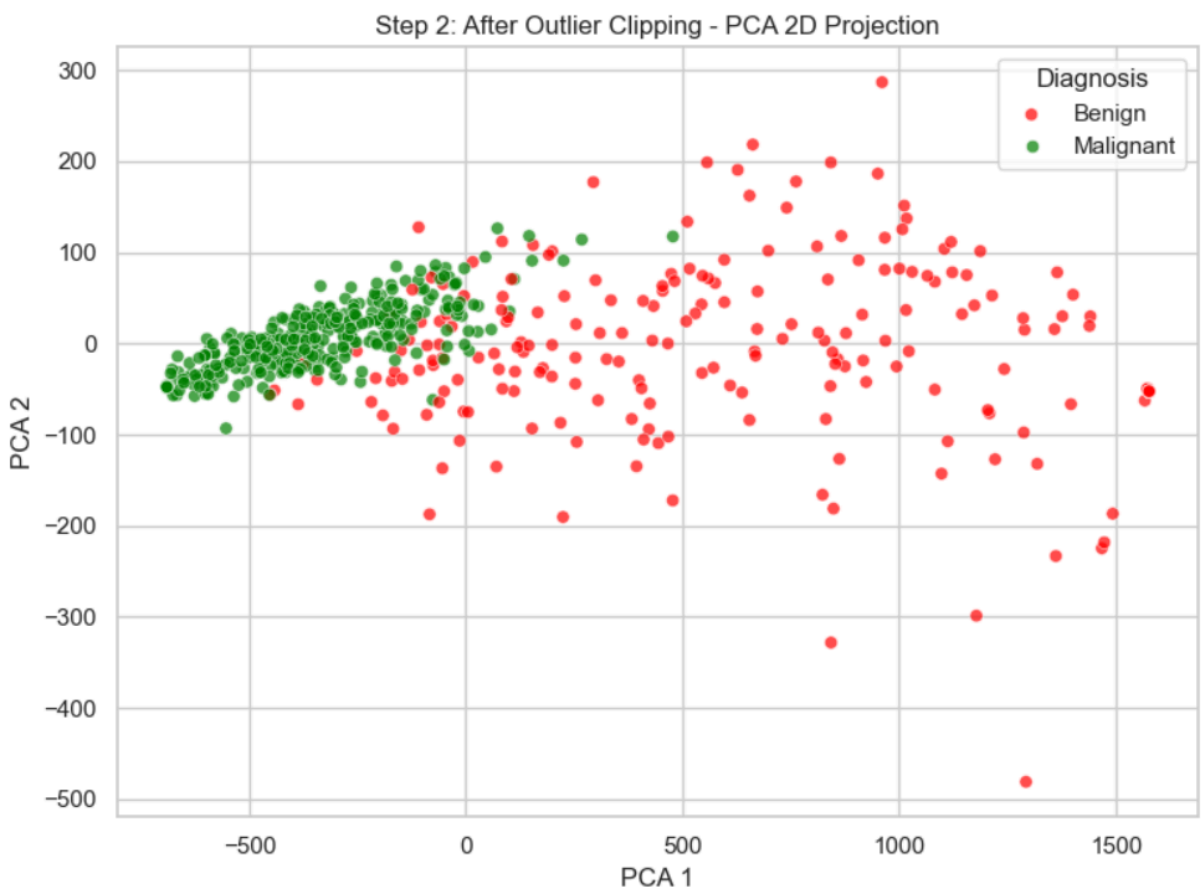
The target variable, diagnosis, originally consisted of categorical labels: 'M' for malignant and 'B' for benign tumors. For compatibility with machine learning algorithms, this variable was binary-encoded:

- **Malignant ('M') → 1**
- **Benign ('B') → 0**

This conversion enabled effective use of classification models and ensured the consistency of numerical processing across the pipeline.

Outlier Treatment via Quantile Clipping

Outliers can significantly distort model learning, particularly in algorithms sensitive to feature scales and distributions. To mitigate this, outlier values in all numeric features were clipped to lie within the 3rd and 97th percentile bounds using the **Interquartile Range (IQR)** [2] method. This approach preserved the distributional characteristics of the data while reducing the influence of extreme observations. Below you can see the data before and after outlier clipping:



Correction of Feature Skewness

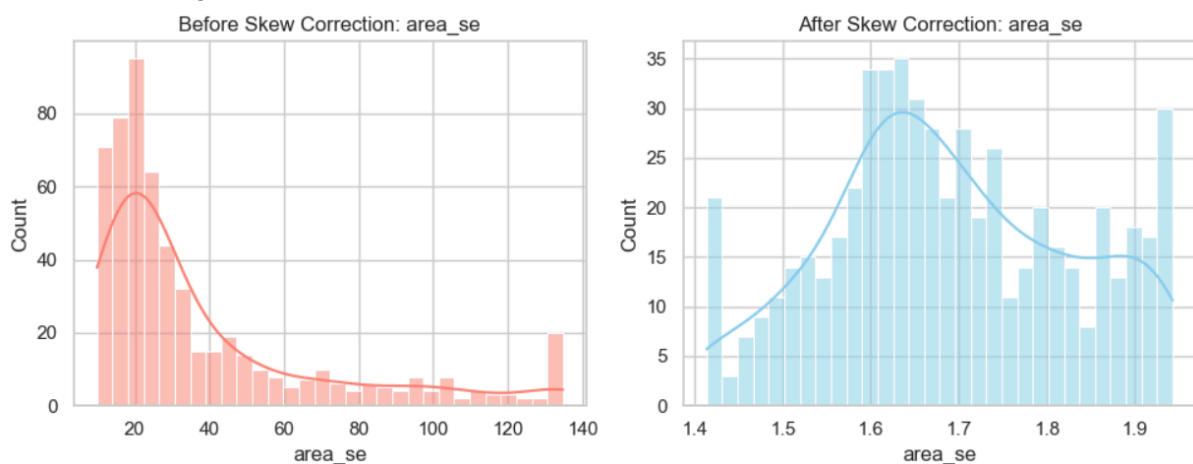
Numerous features in the dataset exhibited high skewness, which can adversely affect model convergence and interpretability. Features with an absolute skewness greater than 1.0 were identified and transformed using appropriate techniques:

- For features with strictly positive values, the **Box-Cox transformation** [3] was applied to approximate normality.
- For non-positive distributions, values were first shifted to a positive range and then log-transformed using **log1p** [4].

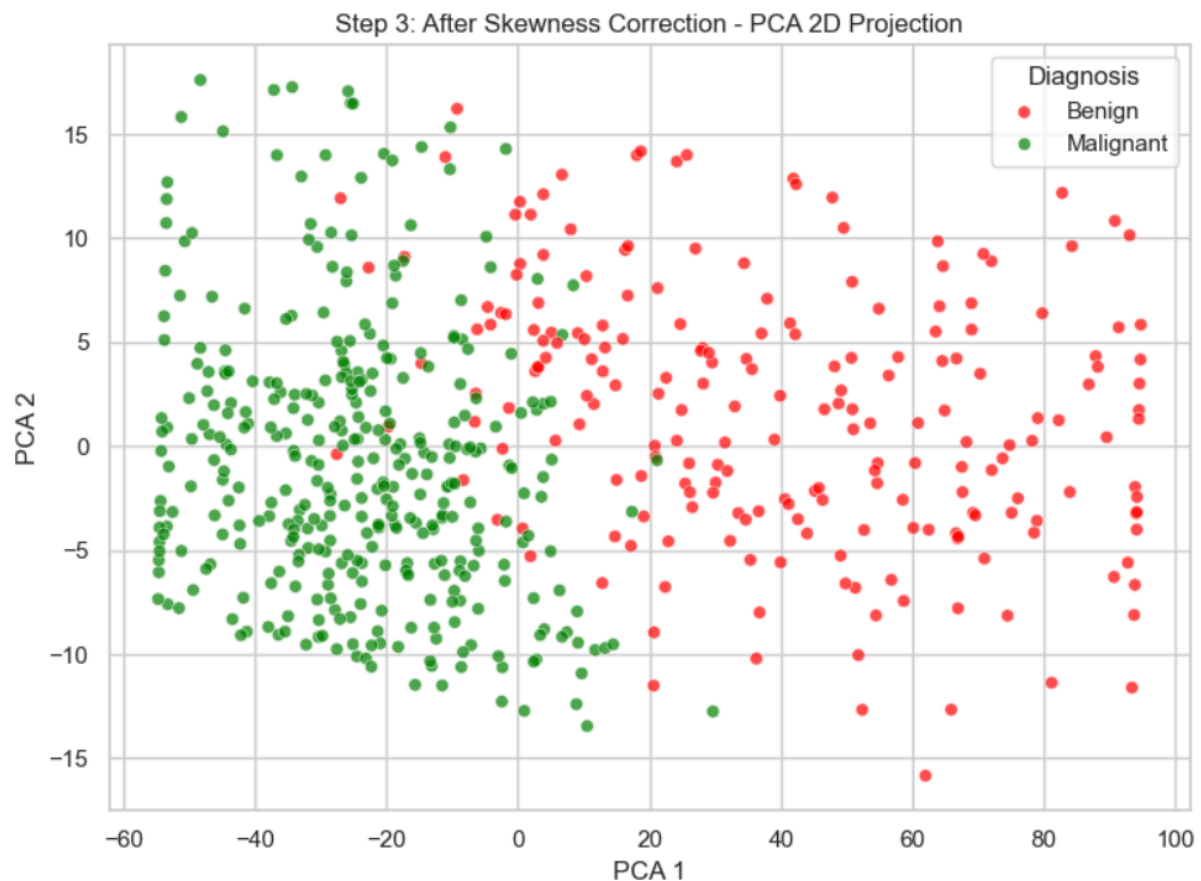
The following features underwent skewness correction:

```
Skewed columns are:  
Index(['area_se', 'perimeter_se', 'radius_se', 'fractal_dimension_se',  
      'area_worst', 'compactness_se', 'symmetry_se', 'concavity_se',  
      'concavity_mean', 'area_mean', 'smoothness_se'],  
      dtype='object')
```

These transformations led to more symmetric distributions, which are favorable for many classification algorithms. Below is an example of skewness correction:



Data after skewness correction:



Feature Scaling

Post skewness correction, all features were standardized using the `StandardScaler` from `scikit-learn`. This transformation centered the features to have a mean of zero and a standard deviation of one, ensuring uniform feature contribution. Standardization is especially critical for distance-based models (e.g., SVM, k-NN) and gradient-based optimizers in neural networks.

```
scaler = StandardScaler()
X_scaled = pd.DataFrame(scaler.fit_transform(X_skew_corrected), columns=X_skew_corrected.columns)
```

Data after scaling:



MODEL TRAINING AND EVALUATION

To accurately classify tumors as benign or malignant, I implemented and evaluated a set of models using both classical machine learning algorithms and deep learning architectures. The goal of this stage was to identify the best-performing model based on key performance metrics: **accuracy, precision, recall, F1-score, and ROC AUC**, using the test set as the benchmark for final evaluation.

Classical Machine Learning Models

I started by training a variety of widely used classical machine learning models. These models were selected for their effectiveness on structured, tabular datasets like the one used in this project. The models I used were:

- **Logistic Regression**
- **Support Vector Classifier (SVC)** with RBF kernel
- **K-Nearest Neighbors (KNN)** with $k=5$
- **Random Forest**

- Gradient Boosting Classifier
- XGBoost

I trained each model on the training set and evaluated its generalization performance using the validation set. After finalizing the models, I evaluated them on the test set. No cross-validation or hyperparameter tuning was applied, as the focus was on building a reliable and interpretable baseline comparison across diverse models.

All models were trained using the **preprocessed feature set**, which included scaled numerical features and binary-encoded diagnosis labels. Performance was tracked using consistent metrics across all data splits to ensure fair comparison.

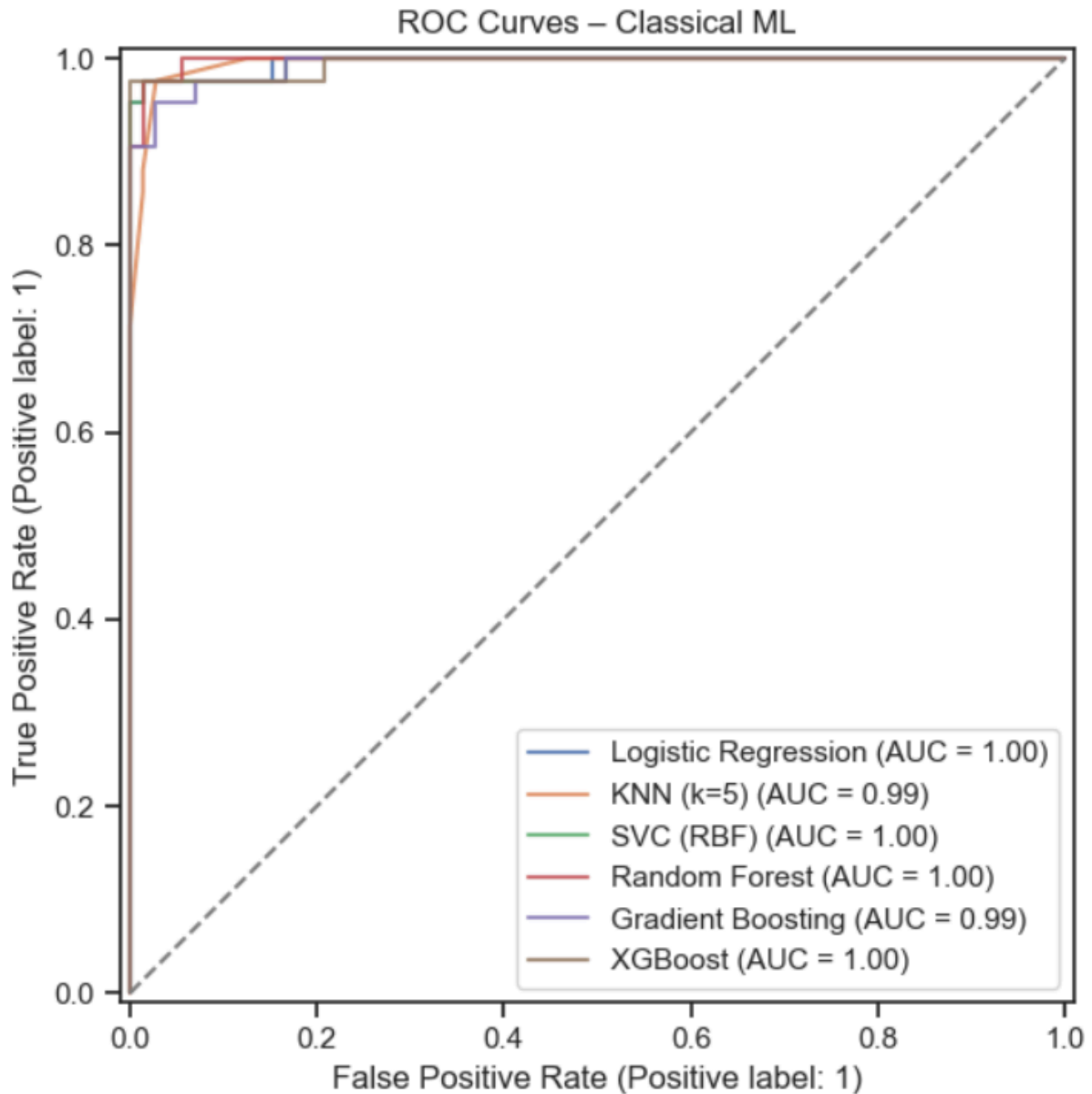
```
for name, model in ml_models.items():
    model.fit(X_train, y_train)
    for split_name, (X_split, y_split) in splits.items():
        y_pred = model.predict(X_split)
        print(f"\n{' '*60}\n{name} [ {split_name} SET\n{' '*60}")
        print(classification_report(y_split, y_pred, target_names=["Benign", "Malignant"], digits=4))

    y_test_pred = model.predict(X_test)
    y_test_proba = model.predict_proba(X_test)[:, 1]

    results.append(dict(
        Model = name,
        Accuracy = accuracy_score(y_test, y_test_pred),
        Precision = precision_score(y_test, y_test_pred),
        Recall = recall_score(y_test, y_test_pred),
        F1 = f1_score(y_test, y_test_pred),
        ROC_AUC = roc_auc_score(y_test, y_test_proba)
    ))

    RocCurveDisplay.from_predictions(y_test, y_test_proba, name=name, ax=plt.gca(), linewidth=1.5, alpha=0.8)

# Save each model
joblib.dump(model, f"../models/{name.replace(' ', '_').replace('(', '').replace(')', '')}.joblib")
```



Deep Learning Models

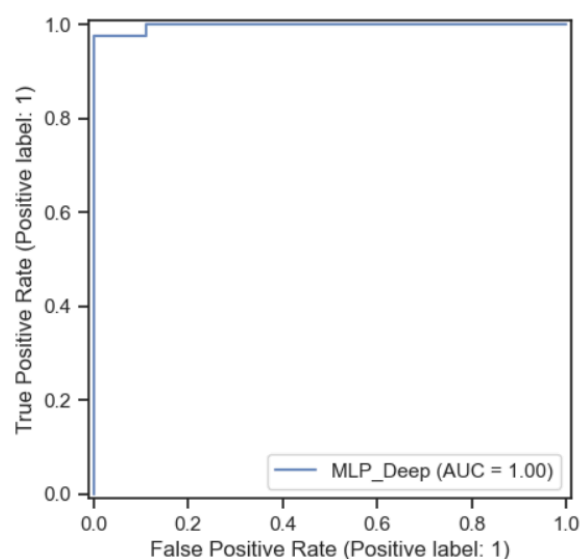
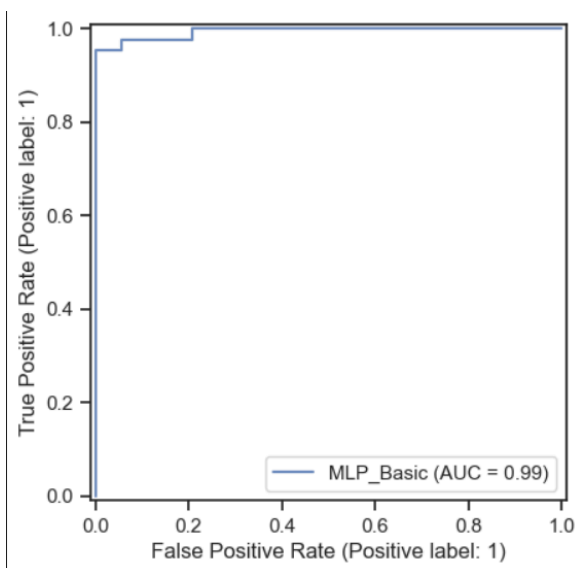
In addition to traditional models, I implemented several **deep learning architectures** using TensorFlow and Keras. These models were designed to explore how neural networks can capture nonlinear relationships in the data. I created three architectures:

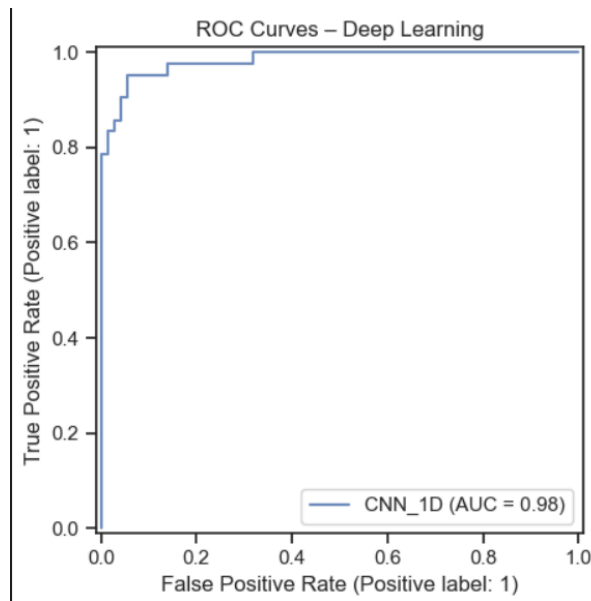
- **MLP_Basic**: A basic multi-layer perceptron consisting of two dense layers with 64 and 32 neurons, ReLU activation, and dropout (0.3 and 0.2) for regularization. This model served as a simple neural baseline.
- **MLP_Deep**: A deeper MLP architecture with three dense layers of 128, 64, and 32 units respectively. It used higher dropout rates to prevent overfitting and was designed to better capture complex patterns in the data.

- **CNN_1D**: A one-dimensional convolutional neural network that first reshaped the feature vector and applied two convolutional layers (32 and 64 filters), followed by max pooling, global average pooling, and dense layers. This model tested whether convolutional architectures could be effective on tabular data.

All deep learning models were compiled using the **Adam optimizer** and **binary cross entropy loss**, with metrics set to **accuracy** and **AUC**. I trained them for up to 100 epochs with **early stopping** based on validation loss (patience = 10), and used a **batch size of 32**. Model checkpoints were not used, but I saved the best weights automatically through early stopping.

```
dl_models = {  
    "MLP_Basic": models.Sequential([  
        layers.Input(shape=(X_train.shape[1],)),  
        layers.Dense(64, activation="relu"), layers.Dropout(0.3),  
        layers.Dense(32, activation="relu"), layers.Dropout(0.2),  
        layers.Dense(1, activation="sigmoid"),  
    ]),  
    "MLP_Deep": models.Sequential([  
        layers.Input(shape=(X_train.shape[1],)),  
        layers.Dense(128, activation="relu"), layers.Dropout(0.4),  
        layers.Dense(64, activation="relu"), layers.Dropout(0.3),  
        layers.Dense(32, activation="relu"), layers.Dropout(0.2),  
        layers.Dense(1, activation="sigmoid")  
    ]),  
    "CNN_1D": models.Sequential([  
        layers.Reshape((X_train.shape[1], 1), input_shape=(X_train.shape[1],)),  
        layers.Conv1D(32, kernel_size=3, activation="relu"), layers.MaxPooling1D(),  
        layers.Conv1D(64, kernel_size=3, activation="relu"), layers.GlobalAveragePooling1D(),  
        layers.Dense(64, activation="relu"), layers.Dropout(0.3),  
        layers.Dense(1, activation="sigmoid")  
    ])  
}
```





Evaluation Results

Model Performance on TEST Set						
Model	Accuracy	Precision	Recall	F1	ROC_AUC	
Random Forest	0.956140	0.974359	0.904762	0.938272	0.998	
MLP_Deep	0.991228	1.000000	0.976190	0.987952	0.997	
Logistic Regression	0.973684	0.953488	0.976190	0.964706	0.996	
SVC (RBF)	0.973684	0.975610	0.952381	0.963855	0.996	
XGBoost	0.973684	1.000000	0.928571	0.962963	0.995	
KNN (k=5)	0.947368	0.973684	0.880952	0.925000	0.995	
MLP_Basic	0.973684	0.975610	0.952381	0.963855	0.994	
Gradient Boosting	0.956140	0.974359	0.904762	0.938272	0.993	
CNN_1D	0.938596	0.926829	0.904762	0.915663	0.983	

The **MLP_Deep** model achieved the **highest overall performance**, with **perfect precision**, excellent recall, and the best F1-score and ROC AUC. This indicates that deeper dense architectures are particularly effective at capturing the underlying patterns in this dataset.

Traditional models like **Logistic Regression**, **SVC**, and **XGBoost** also performed exceptionally well. This demonstrates that even with relatively simple linear or ensemble-based approaches, high accuracy can be achieved when the data is well-preprocessed and features are informative.

The **CNN_1D** model, while still performing well, was outperformed by both MLP architectures and several classical models. This result suggests that convolutional networks may not be the most suitable approach for purely tabular data, where spatial locality is not inherently meaningful.,

STREAMLIT APP DESCRIPTION

Purpose and Functionality Overview

The **Breast Cancer Wisconsin Streamlit App** provides a user-friendly interface for exploring the dataset, comparing model performances, and making predictions. It bridges the gap between complex machine learning models and end-users by enabling interaction with the system through an intuitive web application.

The app serves the following purposes:

- **Demonstrate model performance** across traditional ML and deep learning models.
- **Allow end-users** to make predictions using either pre-uploaded data or manual inputs.
- **Promote understanding** of the dataset with meaningful visualizations.
- **Enable comparison** across different models in terms of accuracy, precision, recall, F1-score, and ROC AUC.

Key Features

Data Upload

Users can upload their own preprocessed .csv files containing feature values. The app handles missing values by filling them with precomputed column-wise means. Uploaded data is used both for visualization and prediction.



Manual Input

If a dataset is not uploaded, users can manually enter values for all 30 numerical features. Each field is pre-filled with the mean value from the training dataset, helping users generate valid test samples easily.

Or manually enter feature values below:

Each input field below is pre-filled with the **average value** from the training dataset. You can change any values to make a prediction based on your own data.

radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
14,1273 - +	19,2896 - +	91,9690 - +	654,8891 - +	0,0964 - +
compactness_mean	concavity_mean	concave points_mean	symmetry_mean	fractal_dimension_mean
0,1043 - +	0,0888 - +	0,0489 - +	0,1812 - +	0,0628 - +
radius_se	texture_se	perimeter_se	area_se	smoothness_se
0,4052 - +	1,2169 - +	2,8661 - +	40,3371 - +	0,0070 - +
compactness_se	concavity_se	concave points_se	symmetry_se	fractal_dimension_se
0,0255 - +	0,0319 - +	0,0118 - +	0,0205 - +	0,0038 - +
radius_worst	texture_worst	perimeter_worst	area_worst	smoothness_worst
16,2692 - +	25,6772 - +	107,2612 - +	880,5831 - +	0,1324 - +
compactness_worst	concavity_worst	concave points_worst	symmetry_worst	fractal_dimension_worst
0,2543 - +	0,2722 - +	0,1146 - +	0,2901 - +	0,0839 - +

Model Selection

Users can:

- Select a model manually from a dropdown menu.
- Automatically select the best or worst model based on various metrics (e.g., F1, ROC AUC).
- View a confusion matrix for the selected model using the validation dataset.

Select model automatically based on metric

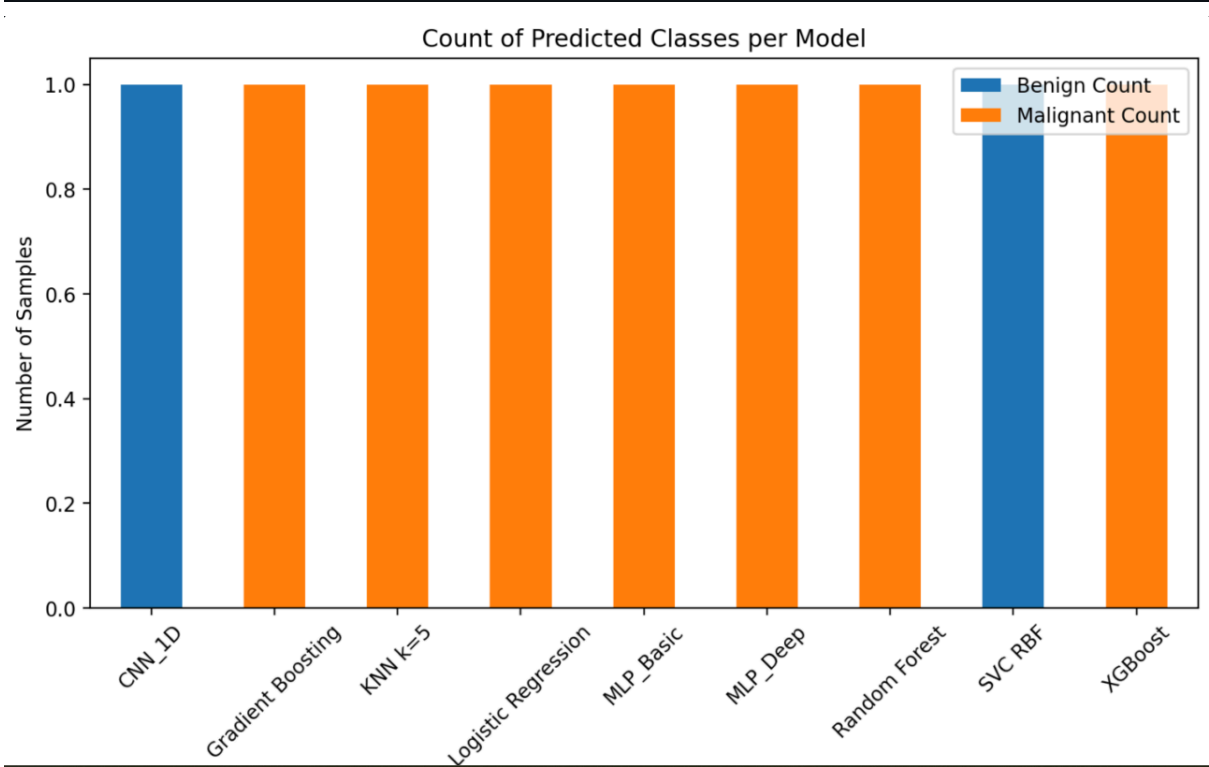
Best Accuracy	Worst Accuracy	Best Precision	Worst Precision	Best Recall	Worst Recall	Best F1	Worst F1	Best ROC_AUC	Worst ROC_AUC
Select a model to inspect / deploy									
Random Forest									

Random Forest
MLP_Deep
Logistic Regression
SVC (RBF)
XGBoost
KNN (k=5)
MLP_Basic
Gradient Boosting
Random Forest

Prediction Visualization

- Users can view predictions and class probabilities in a tabular format.
- The app supports bulk prediction from uploaded files and allows downloading prediction results as .csv.
- When predicting with all models, the app visualizes:
 - Class distribution per model
 - Mean malignancy probabilities per model

Prediction Summary Across Models			
Model	Benign Count	Malignant Count	Mean Malignancy Probability
CNN_1D	1	0	0
Gradient Boosting	0	1	0.9986
KNN k=5	0	1	1
Logistic Regression	0	1	1
MLP_Basic	0	1	1
MLP_Deep	0	1	1
Random Forest	0	1	0.9133
SVC RBF	1	0	0.4791
XGBoost	0	1	0.9975



UI Overview

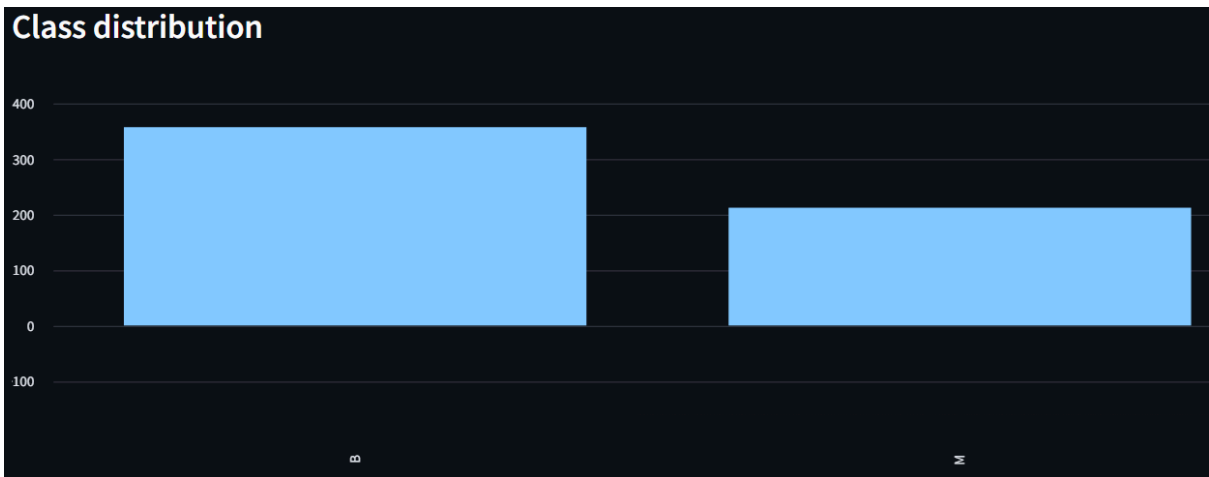
Data Visualisation Page

- Upload dataset
- Show first 5 rows

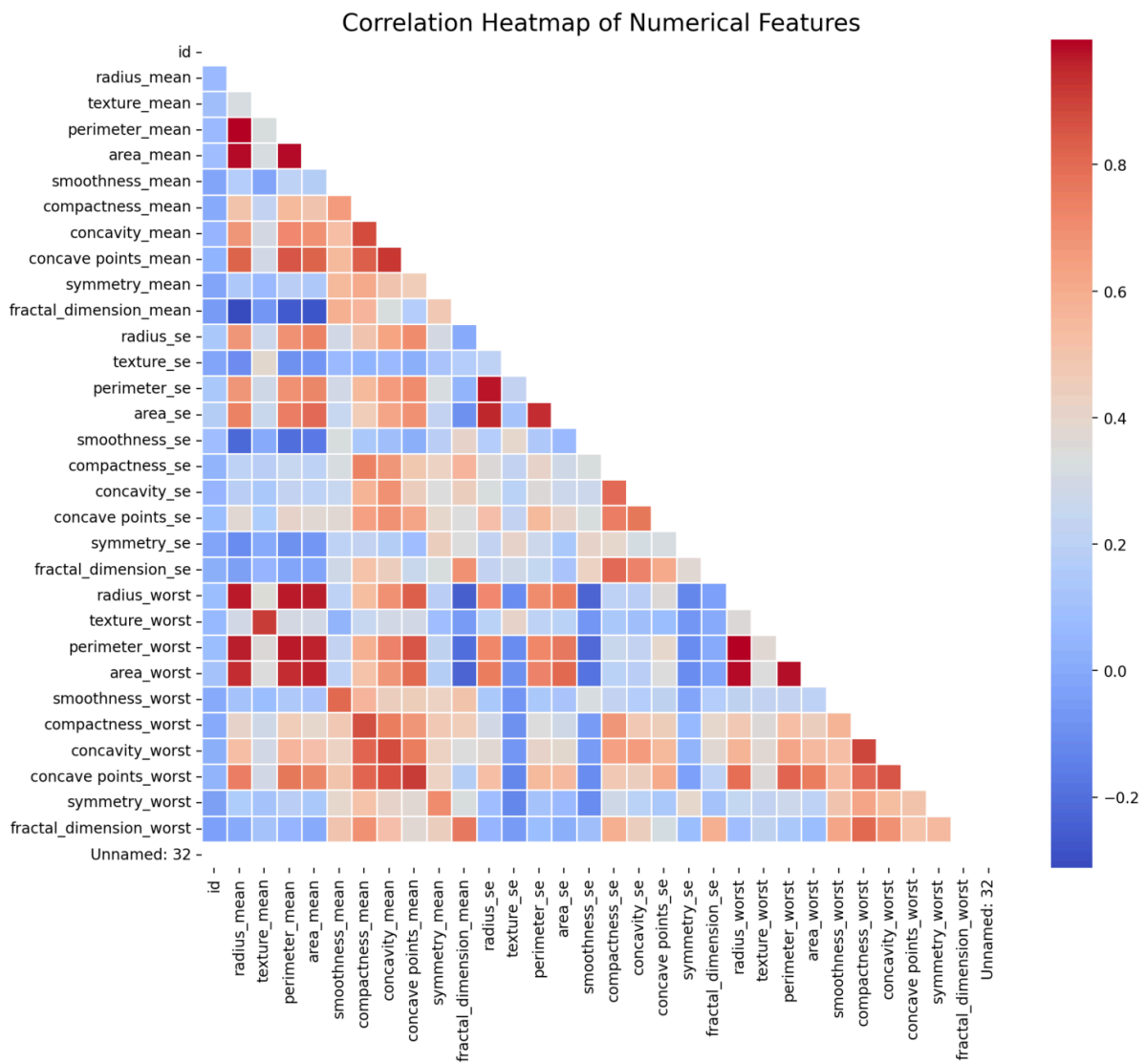
First 5 rows

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave poi
0	842302	M	17.99	10.38	122.8	1001	0.1184	0.2776	0.3001	
1	842517	M	20.57	17.77	132.9	1326	0.0847	0.0786	0.0869	
2	84300903	M	19.69	21.25	130	1203	0.1096	0.1599	0.1974	
3	84348301	M	11.42	20.38	77.58	386.1	0.1425	0.2839	0.2414	
4	84358402	M	20.29	14.34	135.1	1297	0.1003	0.1328	0.198	

- Class distribution bar chart



- Correlation heatmap of numeric features



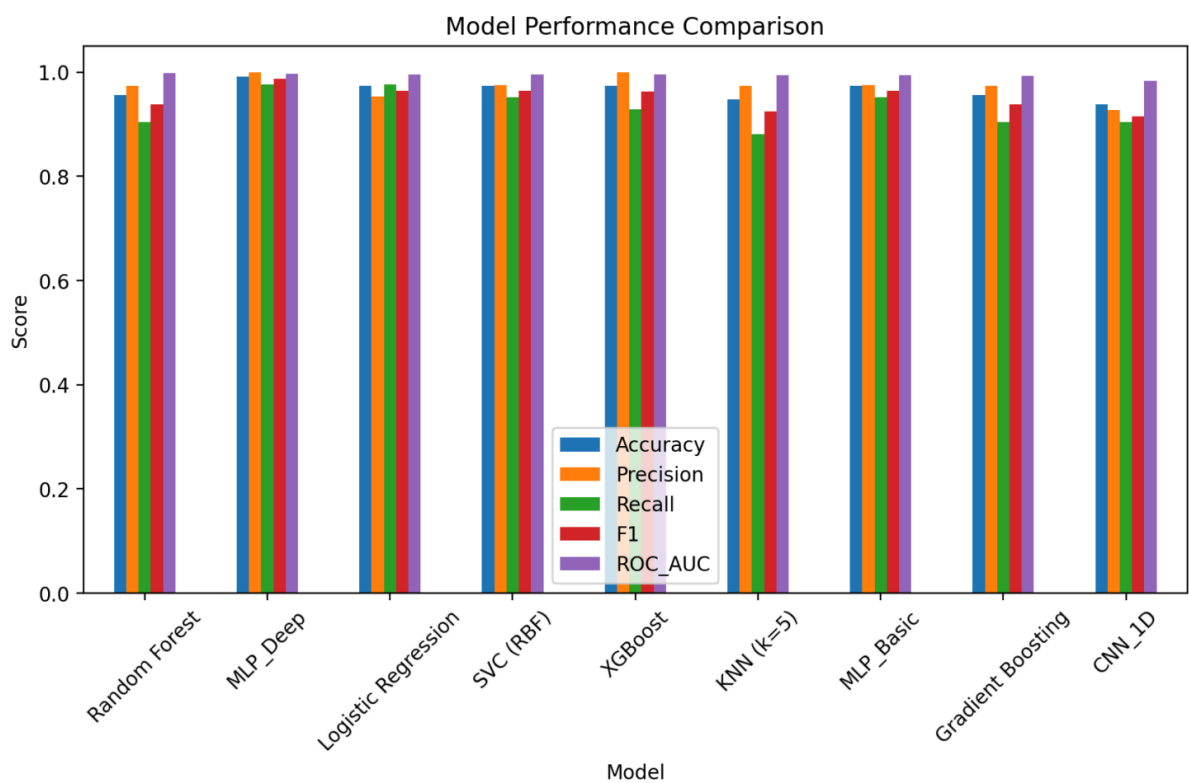
Model Comparison Page

- Model performance metrics table

Compare Trained Models

	Model	Accuracy	Precision	Recall	F1	ROC_AUC
0	Random Forest	0.956	0.974	0.905	0.938	0.998
1	MLP_Deep	0.991	1.000	0.976	0.988	0.997
2	Logistic Regression	0.974	0.953	0.976	0.965	0.996
3	SVC (RBF)	0.974	0.976	0.952	0.964	0.996
4	XGBoost	0.974	1.000	0.929	0.963	0.995
5	KNN (k=5)	0.947	0.974	0.881	0.925	0.995
6	MLP_Basic	0.974	0.976	0.952	0.964	0.994
7	Gradient Boosting	0.956	0.974	0.905	0.938	0.993
8	CNN_1D	0.939	0.927	0.905	0.916	0.983

- Bar charts comparing Accuracy, Precision, Recall, F1, and ROC AUC



- Best/worst model auto-selection buttons

Select model automatically based on metric

Best Accuracy

Worst Accuracy

Best Precision

Worst Precision

Best Recall

Worst Recall

Best F1

Worst F1

Best ROC_AUC

Worst ROC_AUC


Auto-selected model: MLP_Deep

Loaded MLP_Deep

Prediction Page

- CSV upload or manual feature input

Upload CSV

 Drag and drop file here
Limit 200MB per file • CSV

Browse files

Or manually enter feature values below:

Each input field below is pre-filled with the average value from the training dataset. You can change any values to make a prediction based on your own data.

radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
14,1273 - +	19,2896 - +	91,9690 - +	654,8891 - +	0,0964 - +
compactness_mean	concavity_mean	concave points_mean	symmetry_mean	fractal_dimension_mean
0,1043 - +	0,0888 - +	0,0489 - +	0,1812 - +	0,0628 - +
radius_se	texture_se	perimeter_se	area_se	smoothness_se
0,4052 - +	1,2169 - +	2,8661 - +	40,3371 - +	0,0070 - +
compactness_se	concavity_se	concave points_se	symmetry_se	fractal_dimension_se
0,0255 - +	0,0319 - +	0,0118 - +	0,0205 - +	0,0038 - +
radius_worst	texture_worst	perimeter_worst	area_worst	smoothness_worst
16,2692 - +	25,6772 - +	107,2612 - +	880,5831 - +	0,1324 - +

- Predictions with selected model

Predict with selected model

Predictions from CNN_1D

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_r
0	14.1273	19.2896	91.969	654.8891	0.0964	0.1043	0.0888	0.0489	0.

Download predictions

- Option to compare predictions across all models

Predict with all models and compare

Prediction Summary Across Models

Model	Benign Count	Malignant Count	Mean Malignancy Probability
CNN_1D	1	0	0
Gradient Boosting	0	1	0.9986
KNN k=5	0	1	1
Logistic Regression	0	1	1
MLP_Basic	0	1	1
MLP_Deep	0	1	1
Random Forest	0	1	0.9133
SVC RBF	1	0	0.4791
XGBoost	0	1	0.9975

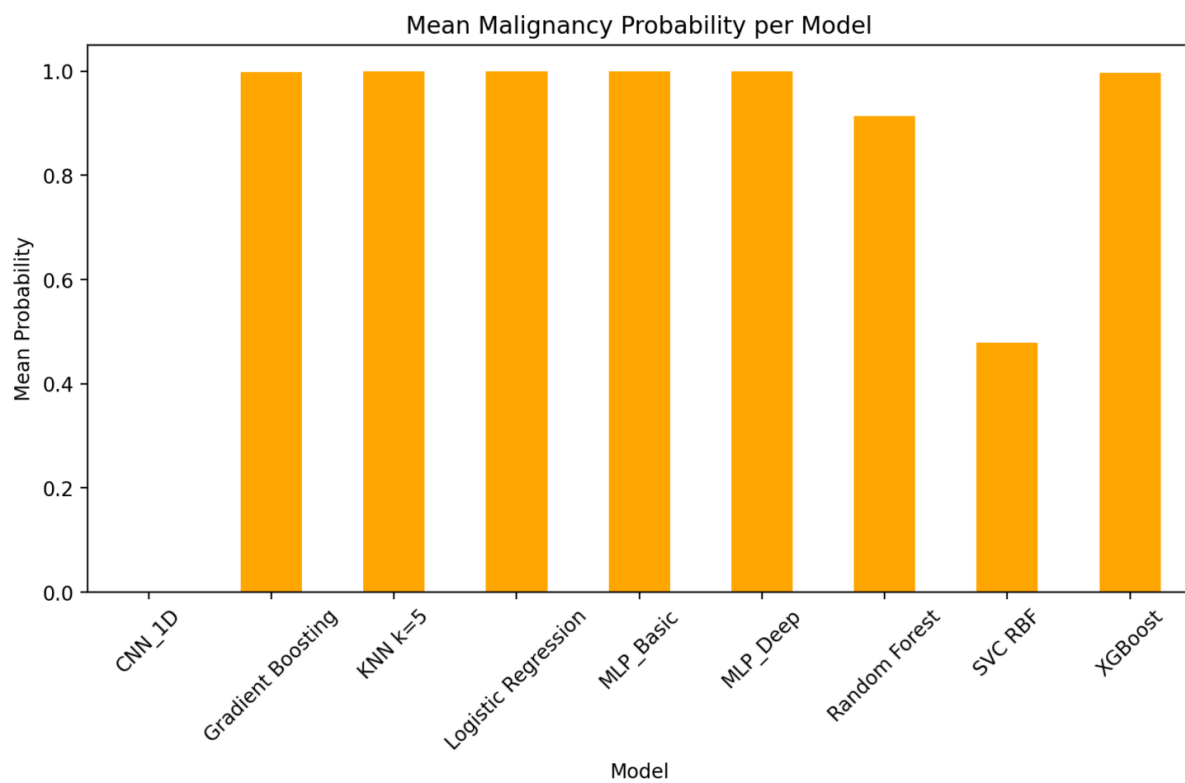
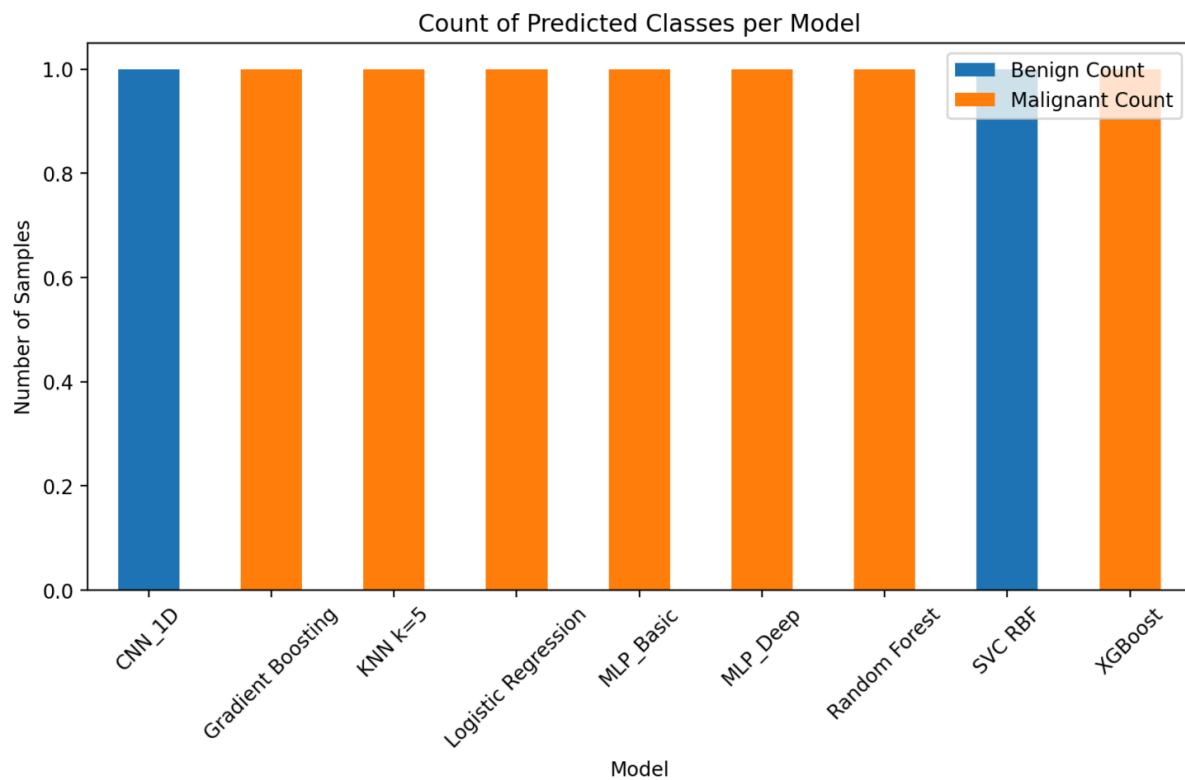
Count of Predicted Classes per Model

Model	Benign Count	Malignant Count
CNN_1D	1	0
Gradient Boosting	0	1
KNN k=5	0	1
Logistic Regression	0	1
MLP_Basic	0	1
MLP_Deep	0	1
Random Forest	0	1
SVC RBF	1	0
XGBoost	0	1

- Downloadable prediction results

Download predictions

- Visualizations of predicted class counts and mean malignancy probabilities



[1] [Breast Cancer Wisconsin \(Diagnostic\) Data Set](#)

[2] Dallah, D., Sulieman, H. (2024). Outlier Detection Using the Range Distribution. In: Kamalov, F., Sivaraj, R., Leung, HH. (eds) Advances in Mathematical Modeling and Scientific Computing. ICRDM 2022. Trends in Mathematics. Birkhäuser, Cham.
https://doi.org/10.1007/978-3-031-41420-6_57

[3] Anthony C. Atkinson. Marco Riani. Aldo Corbellini. "The Box–Cox Transformation: Review and Extensions." *Statist. Sci.* 36 (2) 239 - 255, May 2021.

<https://doi.org/10.1214/20-STS778>

[4] Honglei Zhuang, Xuanhui Wang, Michael Bendersky, and Marc Najork. 2020. Feature Transformation for Neural Ranking Models. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20)*. Association for Computing Machinery, New York, NY, USA, 1649–1652.

<https://doi.org/10.1145/3397271.3401333>