Comparison of Multiple Machine Learning Models for Benign and Malignant Tumor Classification

CID: 06010948

2025-05-04

1. Introduction

1.1 Research Background and Objectives

Promoting the development of new healthcare models such as medical consortia and Internet-based healthcare, along with the application and widespread adoption of digital healthcare technologies, represents a key direction for advancing the healthcare system. Digital medicine leverages computer science and information technology to digitize data, knowledge, and techniques from traditional medical fields. It is widely applied in medical research, clinical diagnosis, and healthcare management, driving the healthcare industry toward greater intelligence and precision.

As a significant branch of artificial intelligence, machine learning excels in data processing and pattern recognition, enabling autonomous learning and dynamic adaptation. Its applications in digital healthcare are increasingly extensive. On one hand, machine learning can efficiently process vast amounts of medical data and clinical cases, enhancing analysis speed and improving physicians' work efficiency. On the other hand, its adaptive nature allows models to continuously optimize based on historical data, thereby improving prediction accuracy and model performance. The role of machine learning in healthcare continues to expand, with widespread applications in personalized treatment support, disease risk assessment, condition analysis, and treatment planning—effectively improving the efficiency and quality of healthcare services.

In recent years, as machine learning models have continued to evolve and improve, their application in clinical disease classification—particularly in distinguishing between benign and malignant tumors—has become a focal point of research in digital medicine. With the help of machine learning, computers can automatically extract tumor-related features from medical imaging data and train classification models to efficiently differentiate between benign and malignant tumors. This approach not only enhances classification accuracy and efficiency but also provides strong decision support for clinicians, aiding in the optimization of treatment plans, improving therapeutic outcomes, and advancing the overall standard and quality of medical services.

This study aims to train multiple machine learning models using tumor-related data to perform predictive classification of tumor malignancy. By comparing the classification performance of different models from multiple perspectives, the objective is to identify the optimal classification method, thereby providing more accurate and efficient technological support for clinical medicine and further enhancing the scientific and precise nature of tumor diagnosis and treatment.

1.2 Review of Related Literature

Researchers have conducted numerous valuable studies in the fields of machine learning, deep learning, and artificial intelligence, significantly advancing the precision and personalization of tumor diagnosis and treatment. Sutton et al. (2020) applied a Recursive Feature Elimination Random Forest (RFE-RF) algorithm based on MRI radiomics classifiers to classify the pathological complete response (pCR) in breast cancer. Their findings demonstrated that the model could accurately predict the pCR status of breast cancer patients following chemotherapy, showing high potential for clinical application. Gokulalakshmi et al. (2020) employed filtering techniques for preprocessing acquired scan images, extracted features using the Gray-Level Co-occurrence Matrix (GLCM) and Discrete Wavelet Transform (DWT) equations, and used Support Vector Machine (SVM) techniques for classification. Their results outperformed traditional brain tumor detection methods in terms of accuracy, recall, and processing time. Maria Sailer et al. (2021) proposed a robust semi-automated end-to-end workflow by constructing a model based on deep convolutional neural networks to predict tumor tissue types in abdominal CT images. Pasha Syed et al. (2022) used SMOTE oversampling to significantly reduce the imbalance in the derived dataset. They further applied classification algorithms such as K-Nearest Neighbors (KNN) and Support Vector Machines (SVM), and also designed a four-layer Artificial Neural Network (ANN) to improve model accuracy. Afrash et al. (2023) trained six machine learning methods—including Multilayer Perceptron (MLP), SVM (with linear and RBF kernels), KNN, Random Forest, and XGBoost—and found that XGBoost performed best in predicting gastric cancer risk.

1.3 Overview of the Report Structure

This study utilizes tumor - related data to train multiple machine learning models, including Support Vector Machine (SVM), Decision Tree, Random Forest, and Adaptive Boosting (Adaboost). By comparing their performances in classifying the malignancy of tumor cells, the aim is to find the optimal method for improving the accuracy of tumor classification. Meanwhile, based on the model results, the development prospects of the application of machine learning models in the medical field are analyzed and discussed, and relevant conclusions and suggestions are given. The specific research content is as follows:

The first part is the introduction, which mainly focuses on the application of machine learning in the classification of benign and malignant tumors. This part first elaborates on the background of the development of the medical system driven by digital medical technologies, highlighting the advantages of machine learning in data processing and pattern recognition

and its extensive application in the field of digital medicine. Furthermore, it explores the potential of machine learning in improving the accuracy and efficiency of tumor classification and assisting in the formulation of treatment plans. Subsequently, it clarifies that this research aims to predict the malignancy of tumors by training multiple models and provide technical support for clinical medicine. Finally, a literature review is carried out to present past research results and application status.

The second part is the dataset description and exploratory data analysis. Firstly, it is introduced that the dataset used in this study is the Breast Cancer Wisconsin (Diagnostic) dataset from the UCI Machine Learning Repository, and the composition and characteristics of the data are explained. Then, data cleaning is carried out, removing columns with missing values, duplicate records, and irrelevant columns, and transforming categorical variables. After that, descriptive statistical analysis is conducted, and significant differences in features between benign and malignant cases are found, with some features showing characteristic distributions. Finally, visual analysis is performed, and the distribution and relationships of key variables are revealed through multiple charts, laying a foundation for subsequent research.

The third part is feature engineering and model construction. On the one hand, feature engineering is carried out. The Min - Max Scaling method is used for feature scaling of the data, and variance filtering and correlation analysis are employed for feature selection to determine 10 key features for modeling. On the other hand, models are constructed. The principles, modeling steps, and relevant formulas of SVM, Decision Tree, Random Forest, and Adaboost are introduced in detail to prepare for model training and evaluation.

The fourth part is model training and evaluation. Firstly, based on the dataset processed by feature engineering, stratified sampling is used to divide the training set and the test set, and a random seed is set to ensure the reproducibility of the experiment. Four models are trained respectively. Secondly, this study conducted feature importance analysis using the Decision Tree and Random Forest models, and identified the most influential variables contributing to tumor malignancy prediction, highlighting key features such as tumor size and boundary irregularity. Then, evaluation metrics such as the confusion matrix, accuracy, macro - precision, macro - recall, and macro F1 - score are used to assess the model performance. The results show that the SVM model performs best, with an accuracy of 97.35% and a macro F1 - score of 97.17%. The performance of the Decision Tree and Adaboost models is similar, and the Random Forest model is slightly inferior. The accuracy of all models exceeds 93%, verifying the effectiveness of feature selection.

The fifth part is conclusions and recommendations. The research shows that the SVM has the best performance in predicting the malignancy of tumor cells. Its unique classification mechanism, good generalization ability, and clear decision - boundary make it suitable for tumor classification tasks. At the same time, machine learning has obvious advantages in the field of tumor classification and can assist in medical decision - making. Based on this, it is recommended to conduct further research from three aspects: model parameter optimization, algorithm fusion and innovation, and dataset expansion and optimization, so as to improve the model performance and provide more powerful support for cancer diagnosis and treatment.

2. Dataset Description

2.1 Data Source

The dataset used in this study is the Breast Cancer Wisconsin (Diagnostic) dataset, sourced from the UCI Machine Learning Repository. It contains diagnostic information and visual characteristics of breast tumor patients. Each record represents an individual patient and includes a unique identifier, tumor diagnosis, and a series of average visual features extracted from medical images—such as radius, texture, perimeter, area, smoothness, compactness, concavity, and concave points. These features are morphological indicators of cell nuclei obtained from fine-needle aspiration (FNA) images, effectively representing tumor morphology and serving as key variables for benign/malignant classification. The dataset is well-structured and clinically relevant, providing a solid foundation for the development and validation of medical diagnostic models. The original dataset is available from the UCI Machine Learning Repository or the Kaggle platform.

2.2 Data Scale

The Breast Cancer Wisconsin (Diagnostic) dataset contains a total of 569 patient samples, each representing a diagnostic record for a breast tumor case. The dataset includes 30 numerical features used for classification, which are morphological indicators of tumor cell nuclei extracted from fine-needle aspiration (FNA) images. In addition, it contains one classification label (diagnosis, where "M" denotes malignant and "B" denotes benign) and one unique identifier (ID). Overall, the dataset is of medium dimensionality and is well-suited for training and evaluating binary classification models.

3. Exploratory Data Analysis

3.1 Data Cleaning

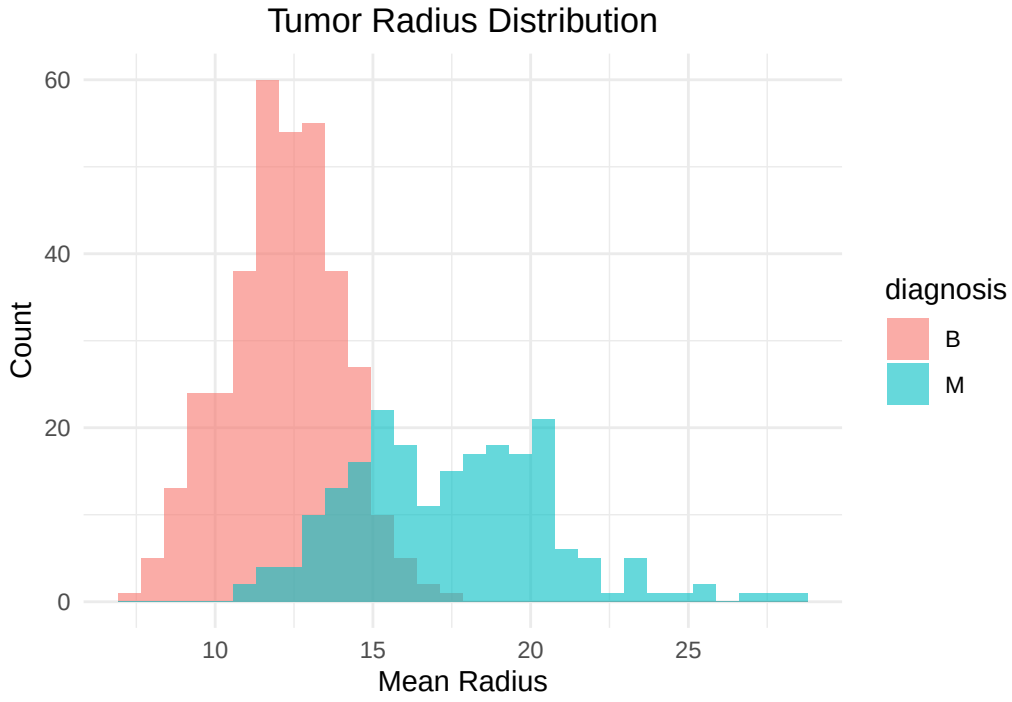
To ensure high-quality data for analysis, we performed preprocessing on the dataset. First, all columns containing only missing values were removed to maintain data integrity. Additionally, duplicate records were identified and deleted to prevent redundancy. The "id" column, which does not contribute to tumor classification, was also eliminated. The "diagnosis" column was converted into a factor variable with two categories: B (Benign) and M (Malignant), allowing for efficient modeling and analysis. After cleaning, the processed dataset was saved as **Cancer_Data_Processed.csv** for further exploration.

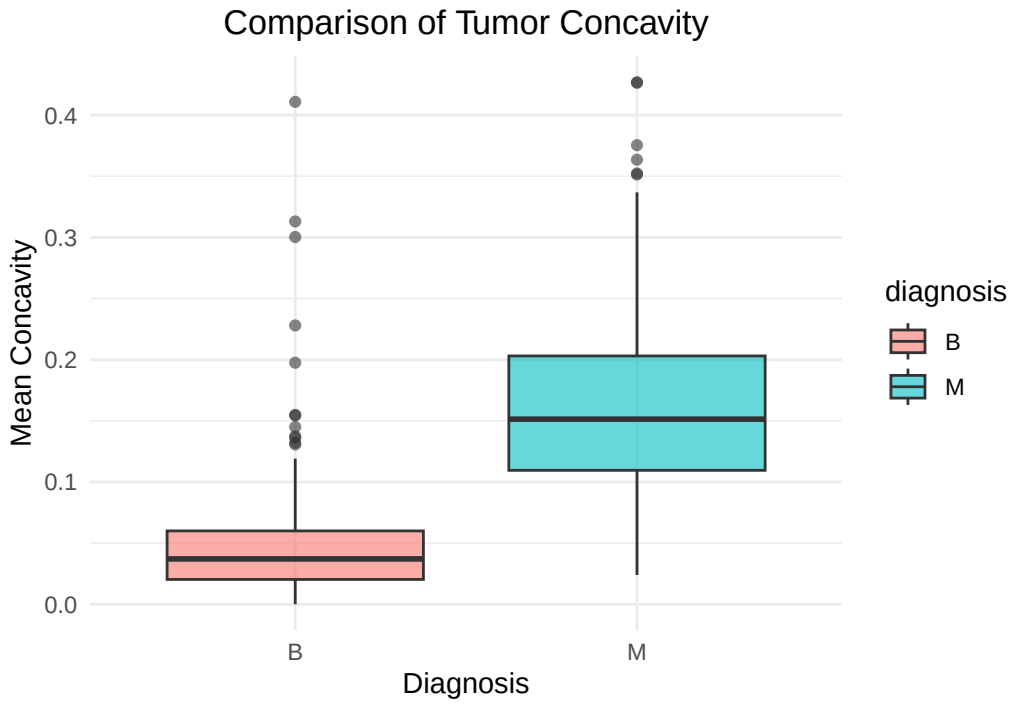
3.2 Descriptive Statistical Analysis

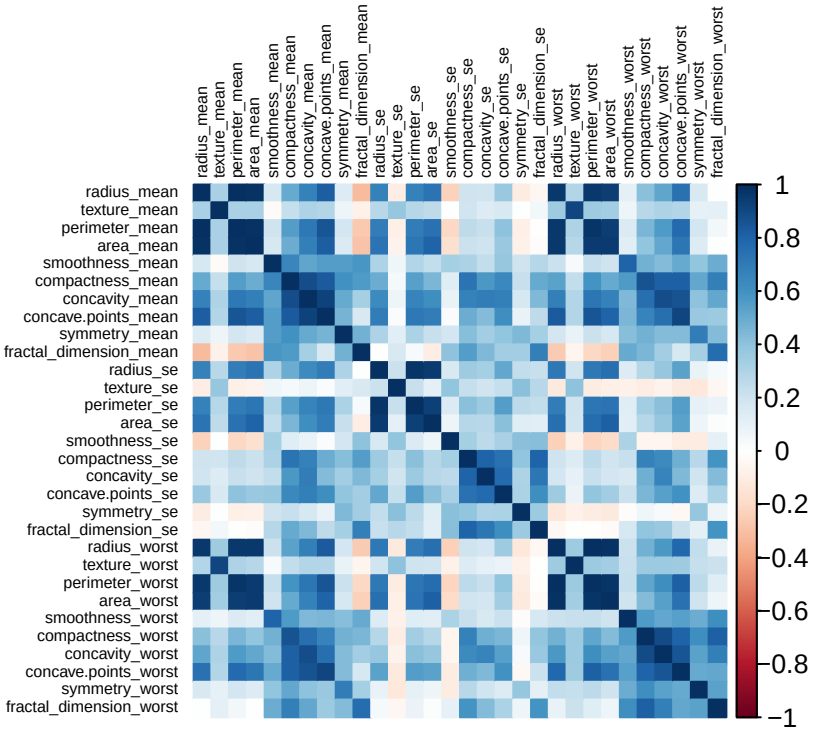
The dataset contains 569 samples, with 357 benign and 212 malignant cases. Summary statistics revealed significant differences between benign and malignant cases, indicating that these features serve as key predictive indicators for subsequent classification models.Most features

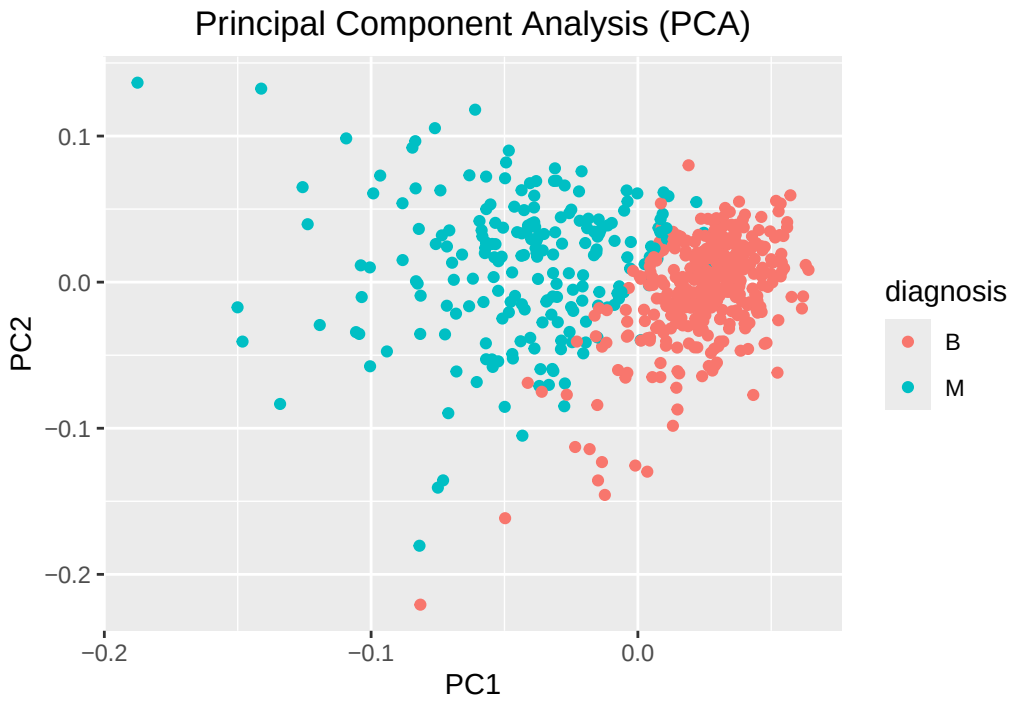
exhibit right-skewed distributions, indicating a concentration of smaller values with occasional extreme observations. Area_mean and area_worst show high skewness and kurtosis, suggesting their importance in tumor classification. Standard error features, particularly concavity_se, have extreme variability, highlighting the potential need for normalization.

3.3 Visual Analysis









In the analysis, we visualized some of the key variables to better understand their distributions and relationships. The histogram of tumor radius revealed that malignant tumors tend to have larger sizes compared to benign ones, reinforcing the importance of tumor size in classification. The box plot of tumor concavity showed significantly higher concavity values for malignant cases, suggesting its relevance in distinguishing tumor types. The correlation

heatmap highlighted strong positive correlations among tumor size-related features (e.g., radius, perimeter, and area), while fractal dimension showed weaker correlations with other attributes, indicating its unique contribution. Finally, the Principal Component Analysis (PCA) scatter plot demonstrated effective dimensionality reduction, with PC1 providing a clear axis of separation between benign and malignant tumors, though some overlap was present. These visualizations enhance our understanding of key predictive features, supporting informed model selection and optimization.

4. Feature Engineering

To improve the training efficiency and classification performance of the model, we carried out systematic feature engineering processing on the data, mainly including two aspects: feature scaling and feature selection.

4.1 Feature Scaling

In the original data, each numerical feature has a different range of values (such as area, radius, texture, etc.). This may cause certain features to have an overly large impact on the model during the training process. To address this issue, we employed the Min - Max Scaling method to scale all numerical variables to the interval [0, 1], enabling each feature to have an equal influence during training. This scaling method helps to enhance the convergence speed and stability of the model while maintaining the relative differences among the original features. It is particularly important for models based on distance or gradient optimization, such as KNN, SVM, and Logistic Regression. The normalized data is saved as **Cancer_Data_Normalized.csv**.

4.2 Feature Selection

Although the original dataset contains 30 numerical features, there is redundancy or weak correlation among some of them. To improve the model's efficiency and generalization ability, we implemented the following feature selection strategies:

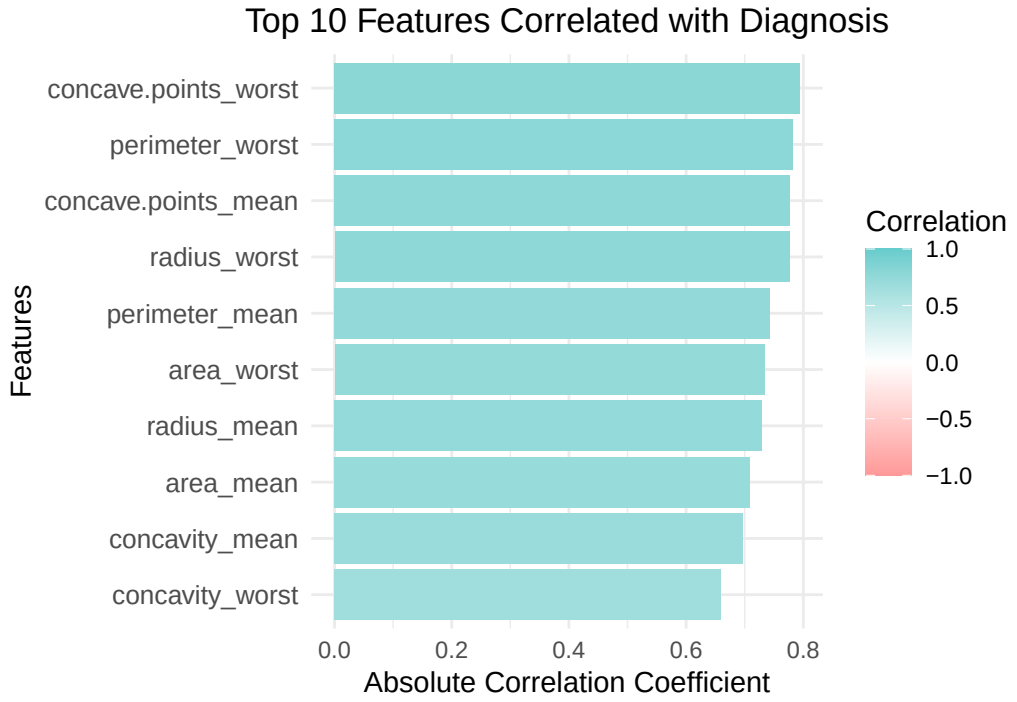
- Variance filtering: Remove variables that hardly change to avoid redundant features.
- Correlation analysis: Calculate the Pearson correlation coefficient between features and the diagnosis label (encoded as 0/1). The formula for the Pearson correlation coefficient (r) is given by:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

  where x_i and y_i are individual data points of two variables, $\bar{x}$ and $\bar{y}$ are the means of those two variables respectively, and n is the number of data points. After calculating the Pearson correlation coefficient, we select the key variables most relevant to the classification task.

The dataset, containing the 10 most important features selected through feature selection along with the response variable, is ultimately used for building machine learning models and saved as **Cancer_Data_Final.csv**.

```
## 'data.frame':    569 obs. of  11 variables:
##  $ concave.points_worst: num  0.912 0.639 0.835 0.885 0.558 ...
##  $ perimeter_worst     : num  0.668 0.54 0.508 0.241 0.507 ...
##  $ concave.points_mean : num  0.731 0.349 0.636 0.523 0.518 ...
##  $ radius_worst        : num  0.621 0.607 0.556 0.248 0.52 ...
##  $ perimeter_mean      : num  0.546 0.616 0.596 0.234 0.631 ...
##  $ area_worst          : num  0.451 0.435 0.375 0.094 0.342 ...
##  $ radius_mean         : num  0.521 0.643 0.601 0.21 0.63 ...
##  $ area_mean           : num  0.364 0.502 0.449 0.103 0.489 ...
##  $ concavity_mean      : num  0.703 0.204 0.463 0.566 0.464 ...
##  $ concavity_worst     : num  0.569 0.193 0.36 0.549 0.319 ...
##  $ diagnosis           : Factor w/ 2 levels "B","M": 2 2 2 2 2 2 2 2 2 2 ...
```



5. Modeling Method Selection and Model Construction

5.1 Support Vector Machine (SVM)

Support Vector Machine (SVM) is a classification method that aims to find a hyperplane to separate samples into two classes with the maximum margin. It mainly deals with two cases: linearly separable and linearly inseparable. For linearly separable cases, it searches for the optimal classification hyperplane in the original space. In linearly inseparable scenarios, slack

variables are introduced, and non - linear samples are mapped to a high - dimensional space to transform the problem into a linear one for finding the best classification hyperplane. This algorithm can effectively address overfitting and local minimum problems that traditional methods may encounter, and it has strong generalization ability. The specific modeling steps are as follows:

1. Assume that a training set in the feature space is given as:

$$T = \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$$

   where x_i is the sample point, y_i is the label of the i-th instance. When $y_i = 1$, it is a positive example; when $y_i = -1$, it is a negative example.

2. In this study, the Gaussian kernel function is selected. The hyperplane and the corresponding classification decision function are constructed as follows:

$$f(x) = \text{sgn}\left(\sum_{i=1}^{n} \alpha_i y_i K(x_i, x) + b\right)$$

$$K(x_i, x) = \exp\left(-\frac{\|x_i - x\|^2}{2\sigma^2}\right)$$

   The distance from any point x in the sample space to the hyperplane can be written as:

$$d = \frac{\left|\sum_{i=1}^{n} \alpha_i y_i K(x_i, x) + b\right|}{\sqrt{\sum_{i=1}^{n} \alpha_i^2}}$$

   where α_i is the Lagrange multiplier, and K is the kernel function.

3. The constraint conditions are determined as:

$$\sum_{i=1}^{n} \alpha_i y_i = 0, \quad C \geq \alpha_i \geq 0, i = 1, 2, \cdots, n$$

4. Use the Lagrange multiplier method and dual problem transformation to solve the objective function.

5.2 Decision Tree

The Decision Tree is a tree - shaped classifier that successively divides the dataset according to different attributes of the data until the data subset contains only the same - class data or there are no more attributes to divide. Essentially, a decision tree is a recursive function that generates branch nodes of the decision tree according to certain rules for data classification or regression. It is a top - down divide - and - conquer method, always seeking the best (step - by - step optimal) at each step, following the "divide - and - conquer" strategy and pursuing "local optimality".

The most common decision tree models are ID3, CART, and C4.5. The CART decision tree can perform both classification tasks and regression or prediction tasks. Therefore, the CART decision tree model is adopted in this study. The specific modeling steps are as follows:

When CART selects the attribute for division, it is based on the Gini index. When measuring classification, the more categories to be divided, the larger the Gini index. The formula for the Gini index is:

$$\text{Gini}(D) = 1 - \sum_{k=1}^{K} p_k^2$$

where D is the dataset, K is the number of classes in D, and p_k is the proportion of the k-th class in D.

The Gini index of attribute a is:

$$\text{Gini - index}(D, a) = \sum_{v=1}^{V} \frac{|D^v|}{|D|} \text{Gini}(D^v)$$

where V is the number of possible values of attribute a, D^v is the subset of D formed by the samples with the v-th value of attribute a.

The attribute that minimizes the Gini index after division is selected as the optimal division attribute, that is, find a_* that satisfies:

$$a_* = \arg\min_a \text{Gini - index}(D, a)$$

According to the above steps, each division attribute is gradually determined, and a decision tree is generated until the pre - set stop - splitting condition is reached.

5.3 Random Forest

Random Forest is an ensemble algorithm that uses multiple decision trees as weak learners to construct a strong learner. It can handle high - dimensional data without the need for feature selection, and it can also determine the importance of features, with a relatively fast training speed.

The Random Forest algorithm is composed of many decision trees, and there is no correlation between different decision trees. The weak classification decision trees it uses are usually CART algorithms, so Random Forest can be used for both classification and regression prediction tasks. Its core idea is to randomly divide all attributes in the dataset into several groups, select one group of attributes to build a decision tree, repeat the above process multiple times to obtain n (pre - set) decision trees, and then use each decision tree for classification and prediction, and finally make a comprehensive prediction based on the voting results.

The specific modeling steps are as follows:

1. Obtain the classification results of each CART decision tree

- a. Perform bootstrap sampling from the full - sample training set T to obtain a training set with a sample size of n.
- b. Use the training set to build a decision tree. For each node on the tree, repeat step 1 and step 2 until the number of samples in the node reaches the minimum limit value:
  - step 1: Randomly select m ($m < p$) variables from all p random variables.
  - step 2: Select the optimal splitting variable from these m variables to split the node into two child nodes.

2. Voting When predicting a new sample, the classification results of multiple decision trees are obtained, and then "voting" is carried out. That is, count the classification results of each decision tree, and take the classification result with the largest number as the final prediction result. If there are multiple classification results with the same number of votes, randomly select one result.

5.4 Adaboost

The AdaBoost algorithm (Adaptive Boosting) trains weak learners sequentially in a highly adaptive manner and is a typical representative of Boosting ensemble learning algorithms.

The adaptability of AdaBoost is reflected in that the algorithm generates an upper bound (which does not need to be specified) during the process and makes the loss function decrease exponentially.

The specific modeling steps are as follows:

1. Initialize the weight distribution of the training data and perform normalization processing on the weights.
2. Iteratively train weak classifiers. In one iteration, if a sample point is correctly classified, its weight is reduced; otherwise, its weight is increased. After updating the weights, train the next classifier.
3. Combine weak classifiers into a strong classifier. Iterate until the number of misclassifications is lower than the preset value or the number of iterations reaches the specified maximum value. After the iteration ends, increase the weight of weak classifiers with a small classification error rate and reduce the weight of weak classifiers with a large classification error rate, and finally obtain a strong learner.

To prevent the AdaBoost algorithm from overfitting, a regularization term - the weight shrinkage coefficient v of each weak learner can be added to the model. The value range of the weight shrinkage coefficient is $(0, 1]$. A smaller value means that more iterations and more weak learners need to be trained to achieve a certain number of misclassifications or learning effects. The iterative formula for weak learners is:

$$f_m(x) = f_{m-1}(x) + \alpha_m h_m(x)$$

The formula after adding the weight shrinkage coefficient is:

$$f_m(x) = f_{m-1}(x) + v\alpha_m h_m(x)$$

6. Model Training and Evaluation

6.1 Model Training Process

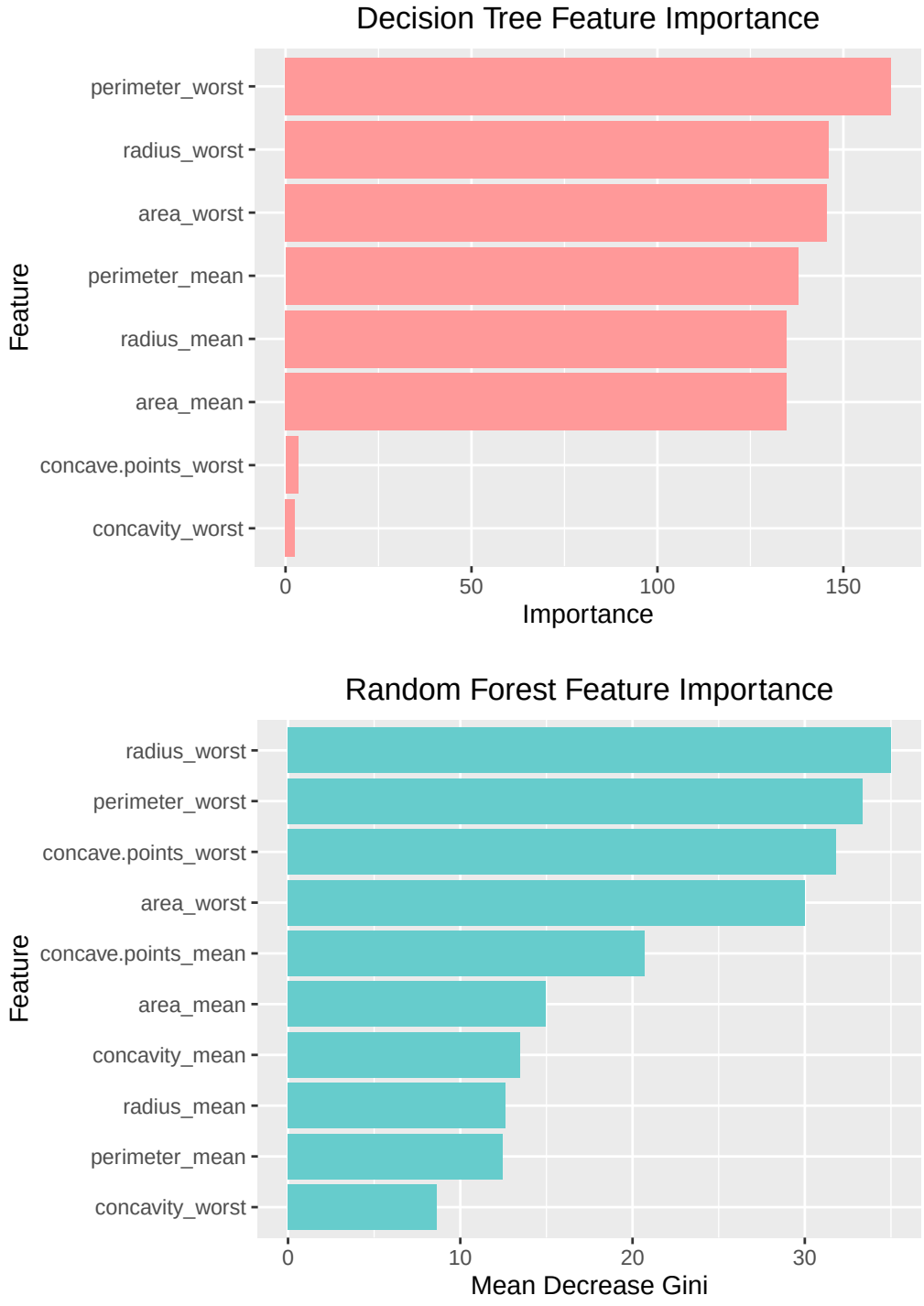
In the model training stage, based on the dataset **Cancer_Data_Final** that has undergone feature engineering processing, this study constructed the above-mentioned four machine learning models, aiming to achieve accurate prediction of the malignancy of tumor cells. A stratified sampling strategy is adopted to divide the dataset. The `createDataPartition` function in the `caret` package is utilized to allocate 80% of the samples as the training set for model parameter learning, and the remaining 20% as the test set for evaluating the model's generalization ability. To ensure the reproducibility of the experiment, the random seed is set to 123.

The specific training processes for each model are as follows:

- **Support Vector Machine (SVM)**: The radial basis kernel function (`svmRadial`) is selected to construct the model. Hyperparameters are optimized through 5-fold cross-validation (`trainControl(method = "cv", number = 5)`). Meanwhile, the training data undergoes centering and standardization preprocessing (`preProcess = c("center", "scale")`), and finally, the optimal classification hyperplane is determined.
- **Decision Tree**: Based on the CART algorithm, with the Gini index as the basis for attribute partitioning. Similarly, a 5-fold cross-validation and data preprocessing strategy are adopted. A tree-shaped classification structure is constructed through recursive partitioning until the preset stopping condition is met.
- **Random Forest**: Using CART decision trees as base learners, multiple decision trees are constructed through bootstrap sampling. When splitting each node, a part of the features is randomly selected to find the optimal splitting variable. Finally, the prediction results of multiple trees are integrated through a voting mechanism to achieve the determination of the tumor cell category.
- **Adaptive Boosting (Adaboost)**: Implemented based on the `gbm` package, weak classifiers are trained iteratively, and the sample weights are dynamically adjusted, assigning higher weights to difficult-to-classify samples. During the training process, 5-fold cross-validation is used to optimize the model, a weight shrinkage coefficient is added to control the risk of overfitting, and a strong classifier is generated by combining weak classifiers with weighted sums.

After the training is completed, each model is used to predict the data in the test set, and the model evaluation indicators are calculated based on the confusion matrix.

6.2 Feature importance analysis



The analysis of the feature importance of decision tree and random forest models reveals that multiple features have a significant impact on the prediction of tumor cell malignancy. Among them, `radius_worst` and `perimeter_worst` hold important positions in both models. A larger worst - case radius and perimeter may indicate that tumor cells have stronger invasiveness and a more aggressive growth pattern, significantly influencing the judgment of

tumor malignancy. `Area_worst` is also crucial as it reflects the spatial extent occupied by tumor cells in a specific state. A larger area may suggest a more advanced stage of tumor development.

In addition, `concave.points_worst` stands out in the random forest model. This feature reflects the irregularity of the tumor cell boundary. A higher number of concave points indicates a more irregular cell morphology, suggesting a greater likelihood of tumor cell malignancy. In the decision tree model, features such as `radius_mean`, `perimeter_mean`, and `area_mean` also show high importance. These mean - value features comprehensively reflect the overall morphological and scale information of tumor cells, providing multi - dimensional references for judging the malignancy of tumors.

6.3 Model Performance Evaluation

After completing the model training and testing, this study systematically evaluates the performance of four models, namely Support Vector Machine (SVM), Decision Tree, Random Forest, and Adaptive Boosting (Adaboost), using indicators such as the confusion matrix, accuracy, macro precision, macro recall, and macro F1-score.

6.3.1 Evaluation Indicators

(1) **Confusion Matrix**: The confusion matrix intuitively presents the classification performance of the model by counting the quantity distribution of the prediction results of each category in the classification task. The rows of the matrix represent the true categories, and the columns represent the predicted categories. The elements on the diagonal represent the number of samples that are correctly classified, while the off-diagonal elements reflect the misclassification situations. By analyzing the confusion matrix, we can clearly identify the misclassification patterns of the model for different categories, providing a basis for model optimization.

(2) **Accuracy**: Accuracy is a fundamental indicator for measuring the overall performance of the model, reflecting the proportion of correctly classified test instances among the total number of test instances. Its calculation formula is:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

where TP (True Positive) is the number of samples that are actually positive and are correctly predicted as positive; TN (True Negative) is the number of true negative samples; FP (False Positive) is the number of false positive samples; and FN (False Negative) is the number of false negative samples. The higher the accuracy, the stronger the model's overall classification correctness. However, in scenarios with imbalanced data categories, this indicator may mask the poor performance of the model for minority classes.

(3) **Macro Precision, Macro Recall, and Macro F1-Measure**:

- **Macro Precision**: Precision measures the proportion of samples that are actually positive among those predicted as positive by the model, reflecting the precision of the model's prediction results. Its calculation formula is $Precision = \frac{TP}{TP+FP}$. Macro precision is the arithmetic average of the precisions of all categories, and the calculation formula is:

$$Macro - Precision = \frac{1}{C}\sum_{i=1}^{C} Precision_i$$

  where C is the total number of categories, and $Precision_i$ is the precision of the i-th category. Macro precision eliminates the influence of differences in the number of samples in each category and can comprehensively evaluate the prediction accuracy of the model for different categories.
- **Macro Recall**: Recall represents the proportion of samples that are actually positive and are correctly predicted as positive, reflecting the model's ability to capture positive samples. The calculation formula is $Recall = \frac{TP}{TP+FN}$. Macro recall is the average of the recalls of all categories, that is:

$$Macro - Recall = \frac{1}{C}\sum_{i=1}^{C} Recall_i$$

  The higher the macro recall, the more balanced the model's ability to identify positive examples across various categories.
- **Macro F1-Score**: The F1-score is the harmonic mean of precision and recall. It comprehensively balances the two through the formula $F1 = \frac{2\times Precision\times Recall}{Precision+Recall}$, avoiding the one-sidedness of a single indicator. The macro F1-score is calculated based on macro precision and macro recall, that is:

$$Macro - F1 = \frac{2 \times Macro - Precision \times Macro - Recall}{Macro - Precision + Macro - Recall}$$

  The closer the macro F1-score is to 1, the better the model's comprehensive classification performance for each category, indicating that it can balance prediction accuracy and completeness.

6.3.2 Comparative Analysis of Model Performance

```
##           Model Accuracy Macro_Precision Macro_Recall Macro_F1
## 1           SVM   0.9735          0.9696       0.9740   0.9717
## 2 Decision Tree   0.9469          0.9476       0.9383   0.9426
## 3 Random Forest   0.9381          0.9297       0.9410   0.9346
## 4      Adaboost   0.9469          0.9382       0.9529   0.9442
```

In the breast cancer diagnosis classification task, the Support Vector Machine (SVM) demonstrated the best performance, achieving a test set accuracy of 97.35% and a macro-average F1-score of 97.17%, indicating its effectiveness in learning nonlinear patterns in the data

through Gaussian kernel mapping. Both the Decision Tree (94.69% accuracy, 94.26% F1-score) and Adaboost (94.69% accuracy, 94.42% F1-score) showed comparable performance, though Adaboost exhibited slightly better recall (95.29%), reflecting its ensemble strategy's enhanced ability to identify minority class samples. The Random Forest (93.81% accuracy, 93.46% F1-score) performed marginally worse, likely due to suboptimal default hyperparameters or weaker feature interactions. Overall, all models achieved accuracies exceeding 93%, validating the effectiveness of the feature selection process. However, the superior performance of SVM suggests it should be prioritized for medical diagnostic tasks, while ensemble methods could potentially be improved through hyperparameter tuning.

7. Conclusions & Recommendations

7.1 Conclusions

In this study, four machine learning models, namely Support Vector Machine (SVM), Decision Tree, Random Forest, and Adaptive Boosting (Adaboost), were constructed and compared to evaluate their performance in predicting the malignancy of tumor cells. The experimental results show that the Support Vector Machine (SVM) model performs best overall. The SVM model classifies samples by finding the optimal hyperplane. When dealing with linearly inseparable data, it maps the data into a higher-dimensional space through kernel functions, thus achieving linear separability. This unique classification mechanism endows SVM with great advantages in predicting the malignancy of tumor cells. For tumor classification tasks, SVM can effectively handle high-dimensional and complex tumor data features. By maximizing the margin between categories, it improves the reliability of classification decisions and reduces the risk of misclassification.

Meanwhile, the SVM model has excellent generalization ability. During the training process, it optimizes the objective function based on the principle of structural risk minimization, avoiding the problem of overfitting. This enables the model to maintain high prediction accuracy when facing new samples. In addition, the decision boundary of SVM is clear and intuitive, which, to a certain extent, interprets the classification logic of the model. This provides convenience for doctors to understand the decision-making process of the model and helps enhance the credibility of clinical applications.

Machine learning methods demonstrate significant advantages and application potential in the field of tumor classification. They can automatically learn from and extract features from massive tumor data, enabling efficient processing of large-scale data and significantly improving the speed of tumor diagnosis and classification. Compared with traditional manual methods, machine learning can reduce the interference of subjective factors, evaluate the malignancy of tumors more comprehensively and objectively, and significantly improve the consistency of classification results. Through in-depth analysis of tumor data and patient information, machine learning can assist doctors in accurately diagnosing and staging tumors, optimizing treatment options, and predicting patient prognosis and treatment effects.

7.2 Recommendations

Based on the results of this study, the following directions for further research and optimization of tumor cell malignancy prediction models are proposed:

1. **Model Parameter Optimization**: Conduct in-depth exploration of parameter adjustment strategies for SVM and other models. For example, optimize the type and parameters of SVM kernel functions (such as the γ value of the radial basis kernel function) and the penalty factor C. Through fine-tuning, further unleash the model's performance and enhance its accuracy and stability in tumor prediction tasks.
2. **Algorithm Fusion and Innovation**: Explore the combination of SVM with other machine learning algorithms or deep learning models to develop hybrid model frameworks. For instance, combine the powerful feature extraction capabilities of neural networks with the classification advantages of SVM, or construct ensemble models with multiple base learners. By complementing each other's strengths, break through the performance bottlenecks of single algorithms and achieve further improvements in prediction accuracy.
3. **Dataset Expansion and Optimization**: Expand the scale of cancer cell sample collection, covering more types of cancer, pathological stages, and patient individual characteristics to ensure data diversity and representativeness. At the same time, strictly control data quality, reducing noise interference through data cleaning, standardization, and other preprocessing methods. In addition, integrate multi-dimensional clinical and biological data (such as gene expression data, proteomics data, and imaging features) to provide richer information inputs for model training, enhancing the model's generalization ability and clinical practicality.

In summary, although the SVM model has demonstrated excellent performance in this study, there is still ample room for optimization in tumor malignancy prediction models. Through continuous improvement of model algorithms and enrichment of data resources, it is expected to provide more accurate and reliable technical support for the early diagnosis and personalized treatment of cancer, contributing to improving patient survival rates and quality of life, and promoting innovative development in the field of oncology.

References

- Afrash M.R., Shafiee M., Kazemi-Arpanahi H. (2023). Establishing machine learning models to predict the early risk of gastric cancer based on lifestyle factors. *BMC Gastroenterol*, 23(6).
- Gokulalakshmi A., Karthik S., Karthikeyan N., Kavitha M.S. (2020). ICM-BTD: improved classification model for brain tumor diagnosis using discrete wavelet transform-based feature extraction and SVM classifier. *Soft computing: A fusion of foundations, methodologies and applications*, 24(24).
- Pasha Syed A.R., Anbalagan R., Setlur A.S., et al. (2022). Implementation of ensemble machine learning algorithms on exome datasets for predicting early diagnosis of cancers. *BMC Bioinformatics*, 23(496).
- Sutton E.J., Onishi N., Fehr D.A., et al. (2020). A machine learning model that classifies breast cancer pathologic complete response on MRI post-neoadjuvant chemotherapy. *Breast Cancer Res*, 22(57).
- Sailer M., Schiller F., Falk T., et al. (2021). Prediction of the histopathological tumor type of newly diagnosed liver lesions from standard abdominal computer tomography with a machine-learning classifier based on convolutional neural networks. *Current Directions in Biomedical Engineering*, 7(1).

Appendice

```
knitr::opts_chunk$set(
  echo = FALSE,
  message = FALSE,
  warning = FALSE
)

knitr::knit_hooks$set(plot = function(x, options) {
  paste0('\\begin{center}\n\\includegraphics[width=',
         ifelse(is.null(options$out.width), '0.8\\textwidth', options$out.width),
         ']{', x, '}\n\\end{center}')
})
# 3.1 Data Cleaning

# Load necessary packages
library(tidyverse)
library(moments)
library(corrplot)
library(ggplot2)
```

```r
library(ggfortify)
library(ggcorrplot)

# Read the raw dataset
data_raw <- read.csv("Cancer_Data_Raw.csv")

# Remove columns that are entirely NA (common example: "X" column)
data_processed <- data_raw[, colSums(is.na(data_raw)) < nrow(data_raw)]

# Display dataset structure
str(data_processed)

# Check for missing values (total and per column)
colSums(is.na(data_processed))

# Check for duplicate rows
sum(duplicated(data_processed))

# Remove columns with no analytical significance, such as "id"
data_processed <- data_processed[, !names(data_processed) %in% c("id")]

# Convert "diagnosis" column to factor (B = Benign, M = Malignant)
data_processed$diagnosis <- factor(data_processed$diagnosis, levels = c("B", "M"))

# Save the cleaned dataset
write.csv(data_processed, "Cancer_Data_Processed.csv", row.names = FALSE)


# 3.2 Descriptive Statistical Analysis

# Display summary statistics
summary(data_processed)

# Calculate mean and standard deviation for each feature grouped by diagnosis
data_processed %>%
  group_by(diagnosis) %>%
  summarize(across(where(is.numeric), list(mean = mean, sd = sd)))

# Count of benign and malignant cases
table(data_processed$diagnosis)

# Compute skewness and kurtosis for each numeric feature
skewness_values <- sapply(data_processed[, sapply(data_processed, is.numeric)],
                          skewness)
```

```r
kurtosis_values <- sapply(data_processed[, sapply(data_processed, is.numeric)],
                          kurtosis)

data.frame(Feature = names(skewness_values), Skewness = skewness_values,
           Kurtosis = kurtosis_values)


# 3.3 Visual Analysis

# Histogram: Tumor radius distribution by diagnosis
ggplot(data_processed, aes(x = radius_mean, fill = diagnosis)) +
  geom_histogram(bins = 30, alpha = 0.6, position = "identity") +
  labs(title = "Tumor Radius Distribution", x = "Mean Radius", y = "Count") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))  # Center title

# Boxplot: Comparison of concavity between benign and malignant tumors
ggplot(data_processed, aes(x = diagnosis, y = concavity_mean, fill = diagnosis)) +
  geom_boxplot(alpha = 0.6) +
  labs(title = "Comparison of Tumor Concavity",
       x = "Diagnosis", y = "Mean Concavity") +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))  # Center title

# Correlation heatmap
cor_matrix <- cor(data_processed[, sapply(data_processed, is.numeric)])
corrplot(cor_matrix, method = "color", tl.cex = 0.5, tl.col = "black")

# Principal Component Analysis (PCA)
pca_res <- prcomp(data_processed[, sapply(data_processed, is.numeric)], scale. = TRUE)

# PCA visualization
autoplot(pca_res, data = data_processed, colour = 'diagnosis') +
  labs(title = "Principal Component Analysis (PCA)", x = "PC1", y = "PC2") +
  theme(plot.title = element_text(hjust = 0.5))  # Center title


# 4.1 Feature Scaling

# Define Min-Max Scaling function
min_max_scaling <- function(x) {
  return((x - min(x)) / (max(x) - min(x)))
}
```

```r
# Apply normalization to all numeric columns except "diagnosis"
data_normalized <- data_processed %>%
  mutate(across(where(is.numeric), min_max_scaling))

# Save the normalized dataset
write.csv(data_normalized, "Cancer_Data_Normalized.csv", row.names = FALSE)

# Display summary of normalized data
summary(data_normalized)


# 4.2 Feature Selection

# Convert diagnosis to numeric (B = 0, M = 1) for further analysis
data_temp <- data_normalized
data_temp$diagnosis_numeric <- as.numeric(data_temp$diagnosis) - 1

# 1. Variance Filtering (excluding the diagnosis column for calculation)
num_data <- data_temp %>% select(where(is.numeric), -diagnosis_numeric)
variance <- apply(num_data, 2, var)

# Set threshold to filter out low-variance features (threshold set to 0.01)
low_variance_features <- names(variance[variance < 0.01])
data_filtered <- data_temp %>% select(-all_of(low_variance_features))

# 2. Correlation Analysis (considering only correlation with diagnosis_numeric)
cor_matrix <- cor(data_filtered %>% select(where(is.numeric)))
cor_target <- cor_matrix["diagnosis_numeric", ]
cor_target <- cor_target[!names(cor_target) %in% c("diagnosis_numeric")]

# Select the top 10 features most correlated with diagnosis
top10_features <- sort(abs(cor_target), decreasing = TRUE)[1:10]
selected_features <- names(top10_features)

# Retain these features for modeling
final_data <- data_filtered %>% select(all_of(selected_features), diagnosis)

# Save the final dataset
write.csv(final_data, "Cancer_Data_Final.csv", row.names = FALSE)

# View the structure of the final dataset
str(final_data)

# Draw a correlation bar chart
```

```r
cor_plot_data <- data.frame(
  Feature = names(top10_features),
  Correlation = cor_target[names(top10_features)],
  Absolute = abs(cor_target[names(top10_features)])
) %>%
  arrange(desc(Absolute))

ggplot(cor_plot_data, aes(x = reorder(Feature, Absolute), y = Absolute,
                          fill = Correlation)) +
  geom_bar(stat = "identity") +
  scale_fill_gradient2(low = "#FF9999", mid = "white", high = "#66CCCC",
                       midpoint = 0, limits = c(-1, 1)) +
  labs(title = "Top 10 Features Correlated with Diagnosis",
       x = "Features",
       y = "Absolute Correlation Coefficient",
       fill = "Correlation") +
  coord_flip() +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5),
        axis.text.y = element_text(size = 10),
        legend.position = "right")


# 6.1 Model Training Process

library(caret)
library(e1071)      # SVM
library(rpart)      # Decision Tree
library(randomForest)
library(gbm)        # Adaboost
library(yardstick)  # For precision/recall/F1
library(dplyr)

# Set the random seed
set.seed(123)

# Split the dataset into training and test sets
train_index <- createDataPartition(final_data$diagnosis, p = 0.8, list = FALSE)
train_data <- final_data[train_index, ]
test_data <- final_data[-train_index, ]

# Train the SVM model
svm_model <- train(
  diagnosis ~ .,
```

```
  data = train_data,
  method = "svmRadial",
  trControl = trainControl(method = "cv", number = 5),
  preProcess = c("center", "scale")
)

# Decision Tree model
dt_model <- train(
  diagnosis ~ .,
  data = train_data,
  method = "rpart",
  trControl = trainControl(method = "cv", number = 5),
  preProcess = c("center", "scale")
)

# Random Forest model
rf_model <- train(
  diagnosis ~ .,
  data = train_data,
  method = "rf",
  trControl = trainControl(method = "cv", number = 5),
  preProcess = c("center", "scale")
)

# Adaboost (based on Gradient Boosting)
ada_model <- train(
  diagnosis ~ .,
  data = train_data,
  method = "gbm",
  trControl = trainControl(method = "cv", number = 5),
  preProcess = c("center", "scale"),
  verbose = FALSE
)


# 6.2 Feature importance analysis

# Decision tree feature importance analysis
dt_importance <- dt_model$finalModel$variable.importance
dt_importance_df <- data.frame(Feature = names(dt_importance),
                               Importance = dt_importance)
# Sort by importance
dt_importance_df <- dt_importance_df[order(-dt_importance_df$Importance), ]
```

```r
# Plot the decision tree feature importance graph
ggplot(dt_importance_df, aes(x = reorder(Feature, Importance), y = Importance)) +
  geom_bar(stat = "identity", fill = "#FF9999") +
  labs(title = "Decision Tree Feature Importance",
       x = "Feature", y = "Importance") +
  coord_flip() +
  theme(plot.title = element_text(hjust = 0.5))

# Random forest feature importance analysis
rf_importance <- randomForest::importance(rf_model$finalModel)
rf_importance_df <- data.frame(Feature = rownames(rf_importance),
                               Importance = rf_importance[, 'MeanDecreaseGini'])
# Sort by importance
rf_importance_df <- rf_importance_df[order(-rf_importance_df$Importance), ]

# Plot the random forest feature importance graph
ggplot(rf_importance_df, aes(x = reorder(Feature, Importance), y = Importance)) +
  geom_bar(stat = "identity", fill = "#66CCCC") +
  labs(title = "Random Forest Feature Importance",
       x = "Feature", y = "Mean Decrease Gini") +
  coord_flip() +
  theme(plot.title = element_text(hjust = 0.5))


# 6.3.2 Comparative Analysis of Model Performance

# Define a unified evaluation function
evaluate_model <- function(predictions, true_labels, model_name) {
  # Create a result data frame
  result_df <- data.frame(
    truth = factor(true_labels, levels = c("B", "M")),
    prediction = factor(predictions, levels = c("B", "M"))
  )

  acc <- as.numeric(accuracy(result_df, truth, prediction)[[".estimate"]])
  precision_macro <- as.numeric(precision(result_df, truth, prediction,
                                          estimator = "macro")[[".estimate"]])
  recall_macro <- as.numeric(recall(result_df, truth, prediction,
                                    estimator = "macro")[[".estimate"]])
  f1_macro <- as.numeric(f_meas(result_df, truth, prediction,
                                estimator = "macro")[[".estimate"]])

  return(data.frame(
    Model = model_name,
```

```
    Accuracy = round(acc, 4),
    Macro_Precision = round(precision_macro, 4),
    Macro_Recall = round(recall_macro, 4),
    Macro_F1 = round(f1_macro, 4)
  ))
}

# Make predictions on the test set and evaluate
svm_pred <- predict(svm_model, newdata = test_data)
dt_pred <- predict(dt_model, newdata = test_data)
rf_pred <- predict(rf_model, newdata = test_data)
ada_pred <- predict(ada_model, newdata = test_data)

# Aggregate evaluation results
results <- bind_rows(
  evaluate_model(svm_pred, test_data$diagnosis, "SVM"),
  evaluate_model(dt_pred, test_data$diagnosis, "Decision Tree"),
  evaluate_model(rf_pred, test_data$diagnosis, "Random Forest"),
  evaluate_model(ada_pred, test_data$diagnosis, "Adaboost")
)

# Output the final model performance table
print(results)
```