Breast Cancer Diagnosis Using Logistic Regression and Random Forest: A Study with the Wisconsin Dataset

anonymous

September 2024

1 Introduction

Breast cancer is one of the most popular cancers affecting women globally. For instance, breast cancer was the most common cancer among women in 157 countries out of 185 in 2022 [2]. Early detection plays a crucial role in improving treatment outcomes and survival rates. If screening coverage across Europe reached 100%, an additional 12,434 deaths could be prevented annually [5]. One of the primary challenges in breast cancer diagnosis is accurately distinguishing between benign (non-cancerous) and malignant (cancerous) tumors. In this context, machine learning has emerged as a promising tool to enhance the accuracy and efficiency of breast cancer classification.

In this report, we approach the classification problem of breast cancer using two different machine learning methods: Logistic Regression and Random Forest. The structure of the report is as follows: the problem formulation is presented in Section 2, followed by the methods and model selection in Section 3. The results and conclusion are discussed in Sections 4 and 5, respectively. The final section provides a summary of the overall findings and discusses the limitations of the methods, along with potential improvements.

2 Problem Formulation

The machine learning task at hand is a supervised classification problem where the goal is to predict whether a breast tumor is benign or malignant based on a set of input features. The dataset used for this task is the Breast Cancer Wisconsin (Diagnostic) dataset [4], which consists of 569 data points. Each data point corresponds to a tumor biopsy sample, where measurements of a tumor's cell nuclei and its corresponding diagnosis are provided. These measurements include 30 features that describe the geometry and texture of the cell nuclei, such as radius, texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry, and fractal dimension. The dataset is publicly available at Kaggle via `https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data/data`.

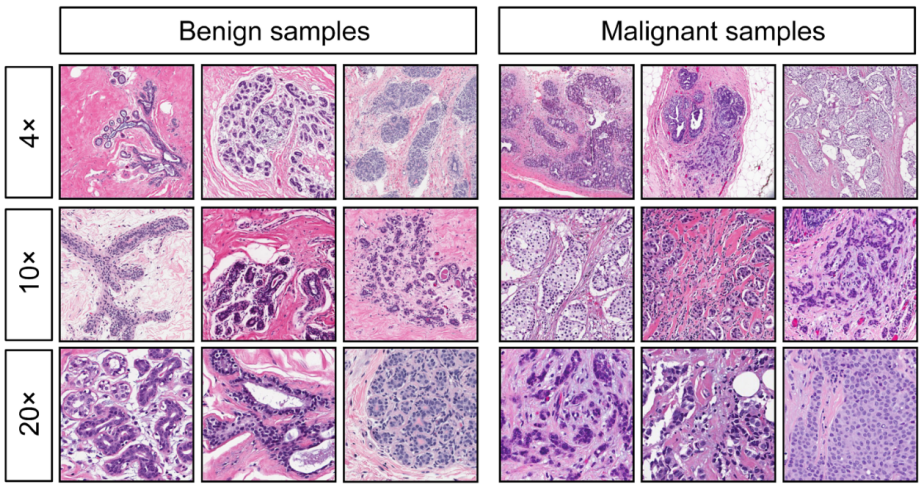


Figure 1: The dataset includes measurements for tumor cells [4]

2.1 Dataset

The dataset $\mathcal{D} = \left\{\left(\mathbf{x}^{(i)}, y^{(i)}\right)\right\}_{i=1}^{m}$ consists of $m = 569$ data points, where each data point $\left(\mathbf{x}^{(i)}, y^{(i)}\right)$ corresponds to a tumor biopsy sample [1]. Specifically, our supervised classification problem can be formulated as follows:

- $\mathbf{x}^{(i)} = [x_1^{(i)}, x_2^{(i)}, \ldots, x_{30}^{(i)}]$ represents a vector of 30 input features describing various characteristics of the tumor.
- In the raw data, the labels are provided as 'M' for malignant and 'B' for benign. However, for the purposes of binary classification, the labels are converted to binary values where $y^{(i)} \in \{0, 1\}$:
  - $y^{(i)} = 0$ indicates that the tumor is benign (originally labeled as 'B').
  - $y^{(i)} = 1$ indicates that the tumor is malignant (originally labeled as 'M').

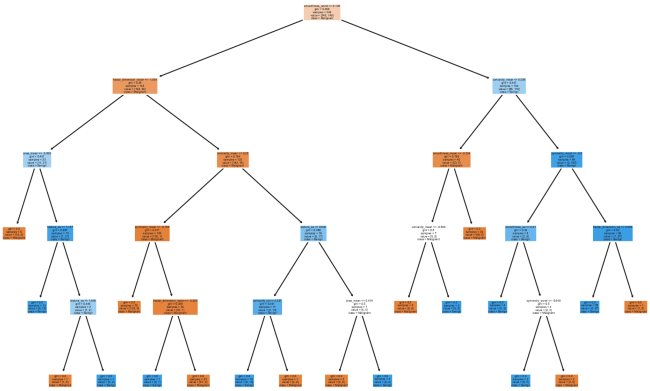


Figure 2: Decision tree

2.2 Features

The dataset contains measurements from FNA biopsies, focusing on the geometry and texture of cell nuclei. These features are categorized into three groups: the mean value, the standard error (SE), and the worst value for each attribute. The mean provides the average measurement of the attribute, SE represents the variation in the measurement, and the worst value captures the most extreme measurement observed. Together, these features provide a comprehensive description of each tumor's characteristics in terms of size, shape, and structural complexity. A summary of these features, along with their meanings and data types, is provided in Table 1. Please see Appendix A. The data type float means that we have continuous variable (except from id and diagnosis, all features are continuous).

3 Methods

3.1 Logistic Regression Model

The first ML model we applied is logistic regression model for this project. Logistic regression is selected because it is applicable for binary classification problems, where the goal is to predict whether a tumor is benign or malignant. One of the key motivations for choosing logistic regression is its interpretability, which is particularly important in medical applications.

In the context of logistic regression, the hypothesis assumes that the log-odds of the probability of the positive class can be expressed as a linear combination of the input features. This linear function is then passed through the logistic (sigmoid) function to output a probability value between 0 and 1.

The model provides coefficients that indicate the contribution of each feature to the prediction. This might help healthcare professionals to understand the factors leading to a model's diagnosis. Logistic regression also assumes a linear relationship between the features and target variable, which is a reasonable assumption for this dataset based on exploratory data analysis.

3.2 Random Forest

As a second method we decided to use decision tree ML method, particularly random forest. This method was selected because it is a powerful ensemble learning technique that combines multiple decision trees to improve classification accuracy. In the context of breast cancer diagnosis, Random Forest can handle complex datasets, capture non-linear relationships, and is robust to overfitting due to its averaging process across multiple trees.

The hypothesis space for Random Forest is defined by an ensemble of decision trees, each constructed from a different bootstrap sample of the training data. The hypothesis space includes all possible decision trees that can be built from the dataset based on different subsets of features and samples.

The model provides feature importance rankings, which can be useful in understanding which factors are most influential in distinguishing between benign and malignant tumors. Its ability to handle imbalanced data and provide balanced error rates makes it ideal for sensitive tasks like cancer detection. The decision tree acquired by applying the test set is represented in the Figure 2.

3.3 Logistic Loss Function and Random Forest Gini

Logistic regression employs binary cross-entropy (logistic loss) as its loss function, which is essential for binary classification tasks. In this project, we choose logistic loss to utilize the `LogisticRegression()` function from the `scikit-learn` library [3].This loss function penalizes incorrect predictions more heavily, motivating the model to adjust its parameters for better accuracy. Formally, logistic regression uses the logistic loss to evaluate the quality of a hypothesis $h^{(w)} \in \mathcal{H}^{(n)}$. Given a labeled $\mathcal{D}$, logistic regression seeks to minimize the empirical risk [1]:

$$\hat{L}(\mathbf{w}|\mathcal{D}) = \frac{1}{m}\sum_{i=1}^{m}\log\left(1+\exp\left(-y^{(i)}h^{(w)}(\mathbf{x}^{(i)})\right)\right)$$

where

$$h^{(w)}(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$$

is the linear combination of the model's parameters $\mathbf{w}$ and the input features $\mathbf{x}$. The overall goal is to find the parameter vector $\hat{\mathbf{w}}$ that minimizes this loss.

Thus, the empirical loss becomes [1]:

$$\hat{L}(\mathbf{w}|\mathcal{D}) = \frac{1}{m}\sum_{i=1}^{m}\log\left(1+\exp\left(-y^{(i)}\mathbf{w}^T\mathbf{x}^{(i)}\right)\right).$$

By minimizing this loss, the model learns to make more accurate predictions, assigning probabilities that reflect the likelihood of each class.

Gini impurity is selected as the criterion for determining the split quality in the random forest model. It is defined mathematically as $1-\sum_i p_i^2$where p_i represents the proportion of data points assigned to the i-th category within a split. The goal of this measure is to create the most balanced splits possible. Both Gini impurity and entropy can achieve this balance, with minimal difference in performance. In a random forest model, multiple decision trees are built, each using a random subset of features and data points, and Gini impurity is applied within each tree to measure split quality. The final prediction is made by aggregating the results from all trees, improving the model's robustness and generalization. Therefore, the default criterion of Gini impurity is used for each tree in this model.

3.4 Training and Validation Split

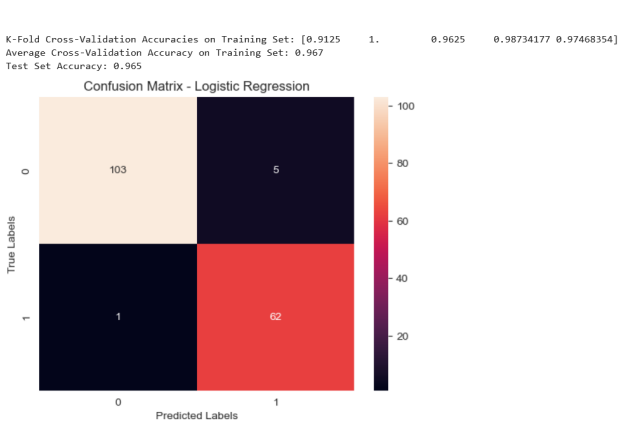
The dataset is split into training, validation, and test sets to ensure the model's performance is evaluated effectively. First, we perform a 70-30 train-test split, where 70% of the data is used for training and validation, and 30% is reserved as the test set for final evaluation. The 70-30 train-test split was chosen to ensure that a substantial portion of the data is allocated for training and validation. This ratio strikes a balance between having enough data to build a robust model and a sufficient amount of unseen data to effectively evaluate the model's performance. For the training process, we apply 5-fold cross-validation on the training set. This method splits the training data into 5 folds, using 4 folds for training and 1 fold for validation, rotating through all the folds. The model is trained on the full training set after cross-validation, and its performance is assessed on the independent test set to estimate its generalization of the model. The overall number of data points is 569. The training set is created using the `train_test_split()`function. After reducing the number of features, as will be described further, the final train set contains 398 data points (test set 171 datapoints) with 13 features.

3.5 Feature Engineering and Selection Process

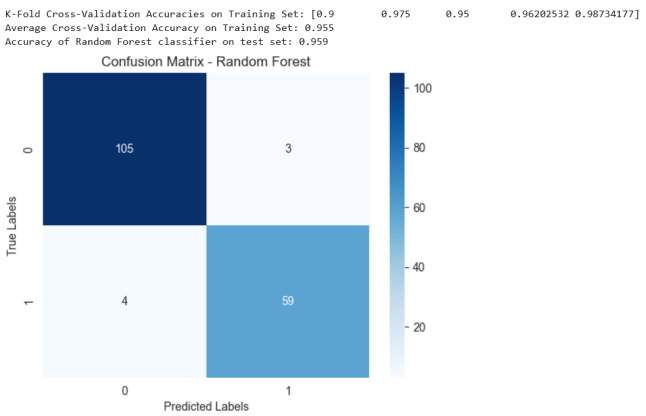
When working with a large dataset for classification, it is crucial to select the most relevant features to avoid overfitting. To eliminate the features that have high correlation the correlation matrix is constructed. Please see Appendix A Fig. 4 and Fig. 5 The data is adjusted using the threshold method. If two features have a high correlation (in this case above 0.8), they contain redundant information. We consider removal of one of these features. Here, we present 4 examples for our case study:

- **Compactness_mean**, **concavity_mean**, and **concave points_mean** exhibit a high degree of correlation. To minimize redundancy, **concavity_mean** is selected as the representative feature.
- **Radius_se**, **perimeter_se**, **area_worst**, **radius_worst**, **perimeter_worst**, **area_mean**, **perimeter_mean**, **radiues_mean**, and **area_se** are also found to be correlated. Among these, **area_mean** is chosen for its relevance.
- Additionally, **compactness_se**, **concavity_se**, and **concave points_se** are correlated. This leads us to the selection of **concavity_se** as the primary feature.
- **Texture_mean** and **texture_worst** are found to be correlated, and **texture_mean** is chosen to represent the texture measurements.

The `StandardScaler()` function is used to standardize the features to have a mean of 0 and a standard deviation of 1. This is important because many machine learning algorithms, such as logistic regression and random forest, can perform better when features are on a similar scale.



(a) Confusion Matrix - Logistic Regression



(b) Confusion Matrix - Random Forest

Figure 3: Comparison of Confusion Matrices

4 Results

Using k-fold cross-validation (5 splits in our case) within the training set eliminates the need for a separate validation set because it allows the model to be both trained and validated on different parts of the training data. In this method, the training data is split into k folds, where the model is trained on k-1 folds and validated on the remaining fold. So in this case we have cross-validation accuracy and general test set accuracy. So, we utilize this method as a two step validation test approach to have a robust model. Both k-fold cross validation averages and general test set result are close Fig. 3, meaning that we have robust models.

The logistic regression model achieves a test accuracy of 96.5% (0.965) and 96.7% (0.967) validation accuracy. As observed from the confusion matrix Figure 3a, the model correctly classifies 103 out of 108 negative instances (true negatives) and 62 out of 63 positive instances (true positives). There are 5 false positives and 1 False Negative, indicating that the model is slightly biased toward the negative class.

Similarly, the random forest model results in a test accuracy of 95.9% (0.959) and 95.5% (0.955) validation accuracy. From its confusion matrix (3b), the model correctly identifies 105 out of 108 negative instances (true negatives) and 59 out of 63 positive instances (true positives). It exhibits 3 false positives and 4 false negatives. This shows a balanced classification with few misses on both classes. The fact that the validation and test accuracy are almost the same indicates that the model is generalizing well to unseen data. This means the model has likely avoided overfitting and is performing consistently across different data sets.

Based on the evaluation criteria and performance metrics, logistic regression method is chosen as the final method. The model outperforms in accuracy the random forest model, additionally given the objective of identifying cancer, where false negatives are more critical than false positives. While both models perform well, logistic regression has a lower number of false negatives (1 compared to 4 in random forest), making it more reliable in identifying true positive cancer cases.

5 Conclusion

In this study, logistic regression and random forest models are assessed for binary classification of cancer detection. Both models exhibits high accuracy, with logistic regression achieving 96.5% and random forest achieving 95.9%. The confusion matrix analysis revealed that logistic regression outperforms random forest by reducing false negatives, which is crucial for identifying positive cancer cases. The logistic regression model correctly identifies 62 out of 63 positive cases, compared to Random Forest's 59 out of 63, indicating its higher reliability in minimizing missed detections.

Despite the strong performance of both models, a key limitation is the balance between false positives and false negatives. While random forest has fewer false positives, the higher number of false negatives makes it less suitable for critical applications like cancer detection, where missing positive cases can have severe consequences. Logistic eegression, with only 1 False Negative, is selected as the final model due to its lower risk of missing a cancer diagnosis.

For future improvements, several strategies can be considered. Techniques such as hyperparameter tuning, the use of ensemble methods, or integrating boosting algorithms may further enhance model performance. Additionally, collecting more data or experimenting with different features can help in addressing any existing model biases and improving the overall classification accuracy.

6 References

[1] A. Jung. *Machine Learning: The Basics*. Singapore: Springer, 2022.

[2] World Health Organization. *Breast Cancer*. `https://www.who.int/news-room/fact-sheets/detail/breast-cancer`. Accessed: 2024-09-15. 2024.

[3] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[4] Mangasarian Olvi Street Nick Wolberg William and W. Street. *Breast Cancer Wisconsin (Diagnostic)*. UCI Machine Learning Repository. 1993.

[5] N. Zielonke et al. "The potential of breast cancer screening in Europe". In: *International Journal of Cancer* 148.2 (2021), pp. 406–418. DOI: `10.1002/ijc.33251`.

7 Appendix A

Table 1: Feature Descriptions in the Wisconsin Breast Cancer Dataset

Feature	**Meaning**	**Data Type**
id	Unique identifier for each instance	categorical
diagnosis	Diagnosis of breast cancer	categorical
radius_mean	Mean of distances from center to perimeter	float
texture_mean	Standard deviation of gray-scale values	float
perimeter_mean	Mean size of the core tumor	float
area_mean	Mean area of the tumor	float
smoothness_mean	Mean of local variation in radius lengths	float
compactness_mean	Mean perimeter^2 / area - 1.0	float
concavity_mean	Mean severity of concave portions of the contour	float
concave_points_mean	Mean number of concave portions of the contour	float
symmetry_mean	Mean symmetry of the tumor	float
fractal_dimension_mean	Mean "coastline approximation" - 1	float
radius_se	SE for the mean of distances from center to perimeter	float
texture_se	SE of gray-scale values	float
perimeter_se	SE for the size of the core tumor	float
area_se	SE for the area of the tumor	float
smoothness_se	SE for local variation in radius lengths	float
compactness_se	SE for perimeter^2 / area - 1.0	float
concavity_se	SE for severity of concave portions of the contour	float
concave_points_se	SE for the number of concave portions of the contour	float
symmetry_se	SE for symmetry of the tumor	float
fractal_dimension_se	SE for "coastline approximation" - 1	float
radius_worst	Largest mean of distances from center to perimeter	float
texture_worst	Largest standard deviation of gray-scale values	float
perimeter_worst	Largest size of the core tumor	float
area_worst	Largest area of the tumor	float
smoothness_worst	Largest local variation in radius lengths	float
compactness_worst	Largest perimeter^2 / area - 1.0	float
concavity_worst	Largest severity of concave portions of the contour	float
concave_points_worst	Largest number of concave portions of the contour	float
symmetry_worst	Largest symmetry of the tumor	float
fractal_dimension_worst	Largest "coastline approximation" - 1	float

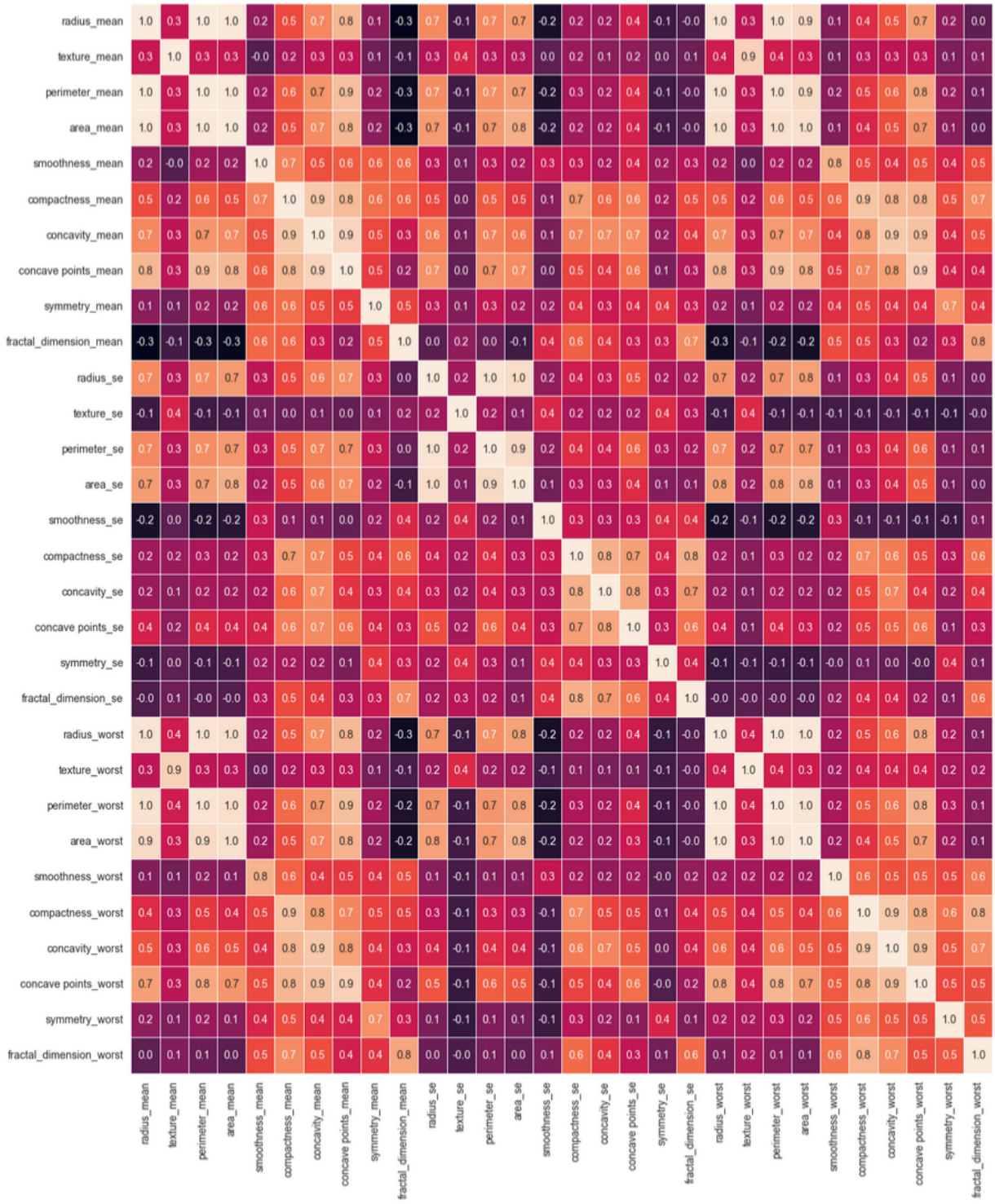


Figure 4: Correlation matrix with all features

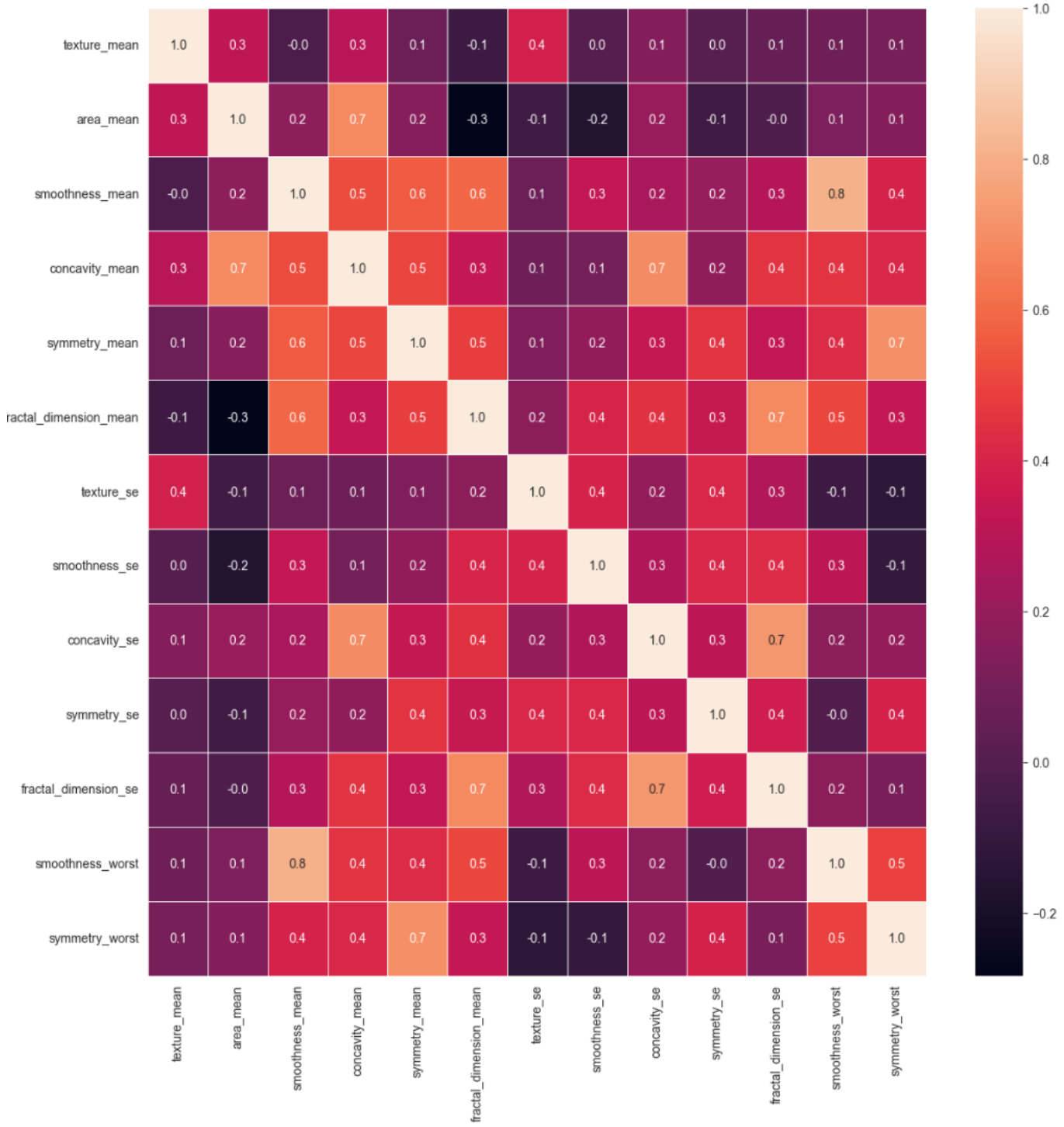


Figure 5: Correlation matrix after eliminating some features

8 Appendix B

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split, KFold, cross_val_score
from sklearn.metrics import f1_score,confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
```

```
df = pd.read_csv('data.csv')
df.head()
```

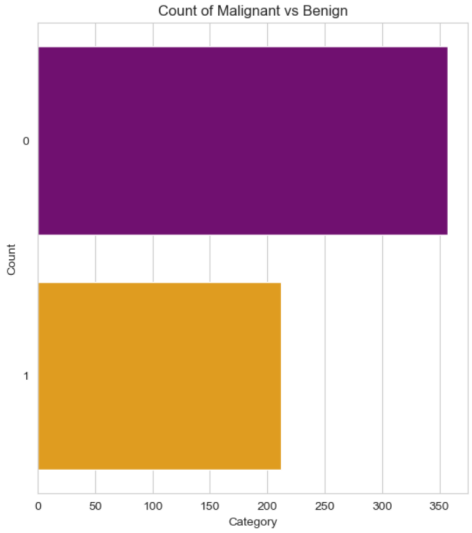
```
col = df.columns
print(col)
```

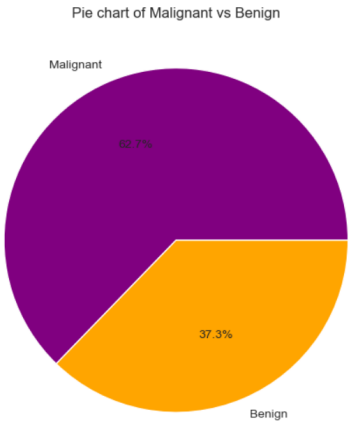
```
# Map M to 1 and B to 0
df['diagnosis'] = df['diagnosis'].map({'M':1,'B':0})
df.head()
```

```
y = df.diagnosis
removed = ['Unnamed: 32','id','diagnosis']
x = df.drop(removed,axis = 1 )
x.head()
```

```
sns.set_style("whitegrid")

fig, ax = plt.subplots(1, 2, figsize=(14, 7))

sns.countplot(y=y, hue=y, ax=ax[0], palette=['purple', 'orange'], legend=False)

counts = y.value_counts()
M, B = counts

print('Number of Malignant: ', M)
print('Number of Benign: ', B)

ax[0].set_title('Count of Malignant vs Benign')
ax[0].set_xlabel('Category')
ax[0].set_ylabel('Count')

# pie
ax[1].pie([M, B], labels=['Malignant', 'Benign'], autopct='%1.1f%%', colors=['
    purple', 'orange'])
ax[1].set_title('Pie chart of Malignant vs Benign')

plt.show()
```

```
Number of Malignant:  357
Number of Benign:  212
```






```
x.describe()
```

```
# batches of ten features
feature_batches = [
    ['radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean', '
       smoothness_mean',
     'compactness_mean', 'concavity_mean', 'concave␣points_mean', 'symmetry_mean'
        , 'fractal_dimension_mean'],
    ['radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
     'compactness_se', 'concavity_se', 'concave␣points_se', 'symmetry_se', '
        fractal_dimension_se'],
    ['radius_worst', 'texture_worst', 'perimeter_worst', 'area_worst', '
       smoothness_worst',
     'compactness_worst', 'concavity_worst', 'concave␣points_worst', '
        symmetry_worst', 'fractal_dimension_worst']
]

custom_palette = {'purple', 'orange'}

for batch_num, batch in enumerate(feature_batches, start=1):
    plt.figure(figsize=(16, 10))
    for i, feature in enumerate(batch, 1):
        plt.subplot(2, 5, i)
        sns.boxplot(data=df, y=feature, hue='diagnosis', palette=custom_palette,
           dodge=True)
        plt.title(f'Feature:␣{feature}', fontsize=12)
    plt.suptitle(f'Box␣Plots␣of␣Features␣-␣Batch␣{batch_num}', fontsize=16)
    plt.tight_layout(rect=[0, 0, 1, 0.95])
    plt.show()
```

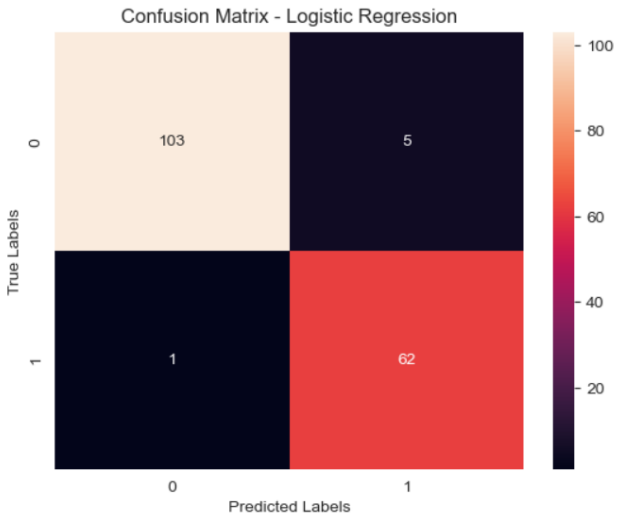
```
# correlation matrix heatmap seaborn

f,ax = plt.subplots(figsize=(18, 18))
sns.heatmap(x.corr(), annot=True, linewidths=.5, fmt= '.1f',ax=ax)
```

```
# Can eliminate features with high correlation, we need to justify these briefly
   in report!!!

eliminated = ['perimeter_mean','radius_mean','area_se','fractal_dimension_worst',
   'compactness_mean','concave␣points_mean','radius_se','perimeter_se','
   concavity_worst','radius_worst','perimeter_worst','compactness_worst','concave
   ␣points_worst','compactness_se','concave␣points_se','texture_worst','
   area_worst']
x_new = x.drop(eliminated,axis = 1 )
x_new.head()
```

```
f,ax = plt.subplots(figsize=(14, 14))
sns.heatmap(x_new.corr(), annot=True, linewidths=.5, fmt= '.1f',ax=ax)
```

```
# Make standardization
scaler = StandardScaler()
x_new = scaler.fit_transform(x_new)

x_train, x_test, y_train, y_test = train_test_split(x_new, y, test_size=0.3,
    random_state=42)

kf = KFold(n_splits=5, shuffle=True, random_state=42)

log_reg = LogisticRegression()

accuracies = cross_val_score(log_reg, x_train, y_train, cv=kf, scoring='accuracy'
    )

print(f'K-Fold Cross-Validation Accuracies on Training Set: {accuracies}')
print(f'Average Cross-Validation Accuracy on Training Set: {accuracies.mean():.3f
    }')

log_reg.fit(x_train, y_train)

y_pred_test = log_reg.predict(x_test)
test_accuracy = accuracy_score(y_test, y_pred_test)

print(f'Test Set Accuracy: {test_accuracy:.3f}')
conf_matrix = confusion_matrix(y_test, y_pred_test)
sns.heatmap(conf_matrix, annot=True, fmt="d")

plt.title('Confusion Matrix - Logistic Regression')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.show()
```

```
K-Fold Cross-Validation Accuracies on Training Set: [0.9125     1.         0.9625     0.98734177 0.97468354]
Average Cross-Validation Accuracy on Training Set: 0.967
Test Set Accuracy: 0.965
```

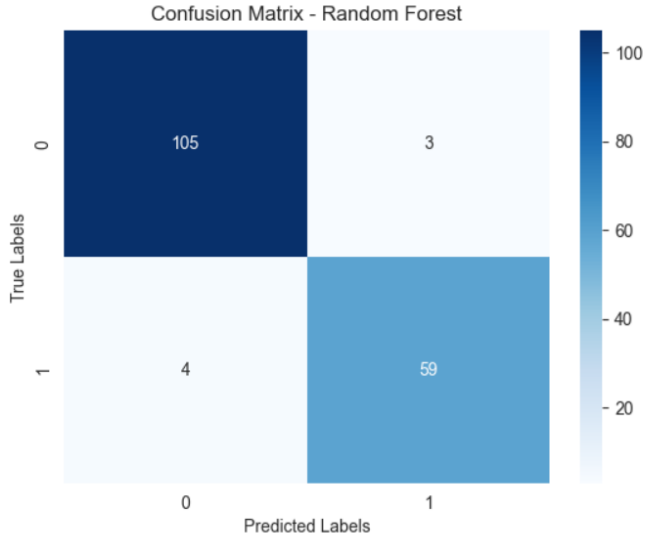



```
# Precision, Recall, F1 Score

print(classification_report(y_test, log_reg.predict(x_test), digits=3))
```

	precision	recall	f1-score	support
0	0.990	0.954	0.972	108
1	0.925	0.984	0.954	63
accuracy			0.965	171
macro avg	0.958	0.969	0.963	171
weighted avg	0.966	0.965	0.965	171

```
kf = KFold(n_splits=5, shuffle=True, random_state=42)

rf = RandomForestClassifier(random_state=42)

rf_cv_accuracies = cross_val_score(rf, x_train, y_train, cv=kf, scoring='accuracy
    ')

print(f'K-Fold␣Cross-Validation␣Accuracies␣on␣Training␣Set:␣{rf_cv_accuracies}')
print(f'Average␣Cross-Validation␣Accuracy␣on␣Training␣Set:␣{rf_cv_accuracies.mean
    ():.3f}')

rf.fit(x_train, y_train)

test_accuracy = rf.score(x_test, y_test)
print(f'Accuracy␣of␣Random␣Forest␣classifier␣on␣test␣set:␣{test_accuracy:.3f}')

conf_matrix = confusion_matrix(y_test, rf.predict(x_test))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues")

plt.title('Confusion␣Matrix␣-␣Random␣Forest')
plt.xlabel('Predicted␣Labels')
plt.ylabel('True␣Labels')
plt.show()
```

```
K-Fold Cross-Validation Accuracies on Training Set: [0.9        0.975      0.95       0.96202532 0.98734177]
Average Cross-Validation Accuracy on Training Set: 0.955
Accuracy of Random Forest classifier on test set: 0.959
```




```
# Recall, Precision, F1 Score

print(classification_report(y_test, rf.predict(x_test), digits=3))
```

	precision	recall	f1-score	support
0	0.963	0.972	0.968	108
1	0.952	0.937	0.944	63
accuracy			0.959	171
macro avg	0.957	0.954	0.956	171
weighted avg	0.959	0.959	0.959	171