Mini Project Report
On
Breast Cancer Prediction using Python

Submitted by

NEEL JAIN 2105806
CHANDANA MISHRA 21051391
ANANYA MISHRA 21051712

School of Computer Engineering
KIIT - DU

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY (KIIT)

Deemed to be University U/S 3 of UGC Act, 1956

Table of Contents

Chapter 1

Introduction

Our project focuses on using machine learning to classify breast cancer tumors as benign or malignant, crucial for early detection and treatment. Leveraging diagnostic images and clinical data, we employ algorithms like Decision Trees, SVM, Naive Bayes, and KNN. Through rigorous preprocessing, training, and evaluation, we aim to identify the most effective model. By providing healthcare professionals with a reliable tool, our project aids in personalized treatment planning and contributes to data-driven healthcare advancements.

- Ananya was responsible for initial data loading, exploration, and basic preprocessing. She loaded the dataset, inspected its shape and basic statistics, and visualized histograms for each variable. Additionally, she also created a correlation heatmap to identify significant correlations between features, laying the foundation for further analysis.

- Chandana handled data preprocessing and initial model evaluation.She split the dataset into predictor variables and target variable, and then divided it into training and testing sets. She defined classification models and evaluated their performance using k-fold cross-validation, establishing baseline performance metrics.

- Neel focused on advanced preprocessing techniques and model prediction on the test set. He standardized the data using StandardScaler for fair comparison among models, repeating model evaluation on standardized data. He made predictions on the test set, evaluated performance metrics, and provided insights into model effectiveness on unseen data, offering valuable insights into model generalization.

Chapter 2

Problem Statement & Objectives

Breast cancer is a significant public health concern worldwide, with early detection being crucial for effective treatment and improved patient outcomes. However, the accuracy and efficiency of current diagnostic methods can be limited, leading to potential misdiagnosis and delays in treatment initiation. There is a need for robust and reliable predictive models that can assist healthcare professionals in accurately classifying breast cancer tumors as benign or malignant based on various diagnostic features.

One selected issue by us is the "Inefficient Diagnosis Process"

Explanation: The current diagnosis process for breast cancer often involves manual interpretation of diagnostic images and clinical data by healthcare professionals. This process can be time-consuming, subjective, and prone to human error, leading to delays in diagnosis and potential misinterpretation of findings. Moreover, the increasing volume of diagnostic data adds to the complexity and workload of healthcare professionals, further impacting the efficiency of the diagnosis process. Addressing this issue is critical to enhancing the accuracy and efficiency of breast cancer diagnosis and improving patient outcomes.

Objectives:

1. Develop machine learning models capable of accurately classifying breast cancer tumors as benign or malignant based on diagnostic features extracted from mammography and biopsy data.

2. Automate and streamline the diagnosis process by integrating the developed machine learning models into clinical workflows, enabling rapid and accurate classification of breast cancer tumors.

3. Evaluate the performance and clinical utility of the developed machine learning models in real-world settings, assessing their impact on diagnostic accuracy, efficiency, and patient outcomes.

Chapter 3

Python Packages Used Details

Name	Functions Used	Explanation
1.numpy	numpy.array()	NumPy is used for numerical computing and provides support for multidimensional arrays and matrices. The numpy.array() function is used to create arrays, which are efficient data structures for numerical computations and manipulation.
2.Pandas	pandas.read_csv() -pandas.DataFrame.describe() -pandas.DataFrame.head() -pandas.DataFrame.drop() -pandas.DataFrame.groupby() -pandas.DataFrame.info()	Pandas is used for data manipulation and analysis. - read_csv() is used to load data from a CSV file into a DataFrame. - describe() generates descriptive statistics of the DataFrame. - head() displays the first few rows of the DataFrame. - drop() removes specified columns or rows from the DataFrame. - groupby() groups the DataFrame based on specified columns. - info() provides a concise summary of the DataFrame, including data types and missing values
3.matplotlib.pyplot	matplotlib.pyplot.plot() - matplotlib.pyplot.show() - matplotlib.pyplot.figure() - matplotlib.pyplot.hist() - matplotlib.pyplot.boxplot()	Matplotlib is used for creating static, interactive, and animated visualizations in Python. Pyplot provides a MATLAB-like interface for creating plots and visualizations. - plot() creates line plots, scatter plots, etc. - show() displays the plot. - figure() creates a new figure for plotting. - hist() generates histograms. - boxplot() creates boxplots to visualize the distribution of data.

4.scikit-learn (sklearn)	sklearn.model_selection.train_test_split() - sklearn.model_selection.cross_val_score() - sklearn.model_selection.KFold() - sklearn.tree.DecisionTreeClassifier() - sklearn.svm.SVC() - sklearn.naive_bayes.GaussianNB() - sklearn.neighbors.KNeighborsClassifier() - sklearn.pipeline.Pipeline() - sklearn.preprocessing.StandardScaler() - sklearn.model_selection.GridSearchCV() - sklearn.metrics.classification_report() - sklearn.metrics.confusion_matrix() - sklearn.metrics.accuracy_score()	Scikit-learn is a machine learning library in Python that provides tools for data preprocessing, model selection, training, evaluation, and more. - train_test_split() is used to split data into training and testing sets. - cross_val_score() performs cross-validation to evaluate model performance. - KFold() splits data into k-folds for cross-validation. - DecisionTreeClassifier(), SVC(), GaussianNB(), and KNeighborsClassifier() are classifiers used for modeling. - Pipeline() is used for chaining together multiple processing steps. - StandardScaler() standardizes features by removing the mean and scaling to unit variance. - GridSearchCV() performs hyperparameter tuning using grid search. - classification_report() generates a classification report including precision, recall, and F1-score. - confusion_matrix() computes confusion matrix to evaluate classification performance. - accuracy_score() computes accuracy of the classification. Sklearn is essential for building and evaluating machine learning models, including preprocessing data, selecting models, tuning hyperparameters, and evaluating model performance.

Chapter 4

Source Code

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, cross_val_score, KFold
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

# Load Data
df = pd.read_csv('data.csv')

# Display first few rows of the dataset
df.head()

# Shape of the Dataset
df.shape

# Descriptive statistics of the dataset
df.describe()

# Set the ID column to be the index of the dataframe
df = df.set_index('id')

# Drop 'Unnamed: 32' column
df.drop(['Unnamed: 32'], axis=1, inplace=True)

# Columns in the dataset
df.columns

# Encode diagnosis column such that M = 1, B = 0
df['diagnosis'] = df['diagnosis'].apply(lambda x: 1 if x == 'M' else 0)

# Display counts of Benign and Malignant cases
print(df.groupby('diagnosis').size())

# Information about the dataset
```

```
df.info()

# Plot histograms for each variable
sns.set_style('darkgrid')
df.hist(figsize=(30, 30))
plt.show()

# Plot heatmap of correlations
plt.figure(figsize=(30, 20))
cor = df.corr()
sns.heatmap(cor, annot=True, cmap=plt.cm.Reds)
plt.show()

# Correlation with output variable
cor_target = abs(cor["diagnosis"])

# Selecting highly correlated features
relevant_features = cor_target[cor_target > 0.7]

# Split the data into predictor variables and target variable, followed by train-test split
Y = df['diagnosis'].values
X = df.drop('diagnosis', axis=1).values
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.20, random_state=21)

# Define models to train
models = [
   ('CART', DecisionTreeClassifier()),
   ('SVM', SVC()),
   ('NB', GaussianNB()),
   ('KNN', KNeighborsClassifier())
]

# Evaluate each model in turn
results = []
names = []

for name, model in models:
   kfold = KFold(n_splits=10)
   cv_results = cross_val_score(model, X_train, Y_train, cv=kfold, scoring='accuracy')
   results.append(cv_results)
   names.append(name)
      msg = "For %s Model: Mean accuracy is %f (Std accuracy is %f)" % (name,
cv_results.mean(), cv_results.std())
   print(msg)

# Compare model performances using boxplots
fig = plt.figure(figsize=(10, 10))
fig.suptitle('Performance Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
```

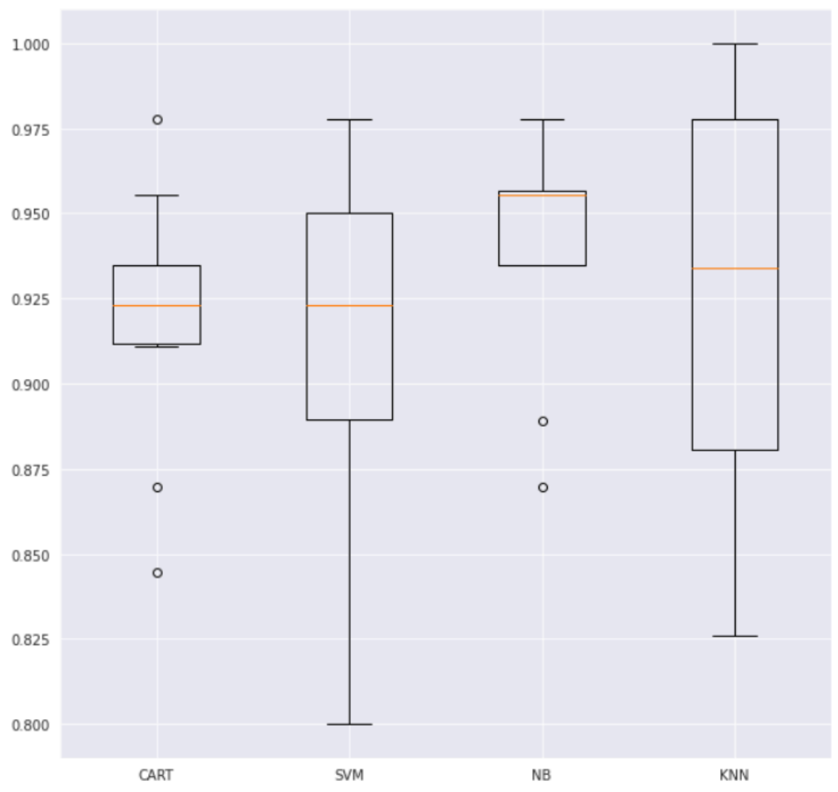
```
ax.set_xticklabels(names)
plt.show()

# Standardize the dataset
pipelines = [
        ('Scaled CART', Pipeline([('Scaler', StandardScaler()), ('CART', DecisionTreeClassifier())])),
  ('Scaled SVM', Pipeline([('Scaler', StandardScaler()), ('SVM', SVC())])),
  ('Scaled NB', Pipeline([('Scaler', StandardScaler()), ('NB', GaussianNB())])),
  ('Scaled KNN', Pipeline([('Scaler', StandardScaler()), ('KNN', KNeighborsClassifier())]))
]

results = []
names = []

kfold = KFold(n_splits=10)
for name, model in pipelines:
  cv_results = cross_val_score(model, X_train, Y_train, cv=kfold, scoring='accuracy')
  results.append(cv_results)
  names.append(name)
     print("For %s Model: Mean Accuracy is %f (Std Accuracy is %f)" % (name, cv_results.mean(), cv_results.std()))

fig = plt.figure(figsize=(10, 10))
fig.suptitle('Performance Comparison For Standardized Data')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()

# Make predictions on validation dataset
for name, model in models:
  scaler = StandardScaler().fit(X_train)
  X_train_scaled = scaler.transform(X_train)
  model.fit(X_train_scaled, Y_train)
  X_test_scaled = scaler.transform(X_test)
  predictions = model.predict(X_test_scaled)
  print("\nModel:", name)
  print("Accuracy score:", accuracy_score(Y_test, predictions))
  print("Classification report:\n", classification_report(Y_test, predictions))
  print("Confusion Matrix:\n", confusion_matrix(Y_test, predictions))
```

Chapter 5

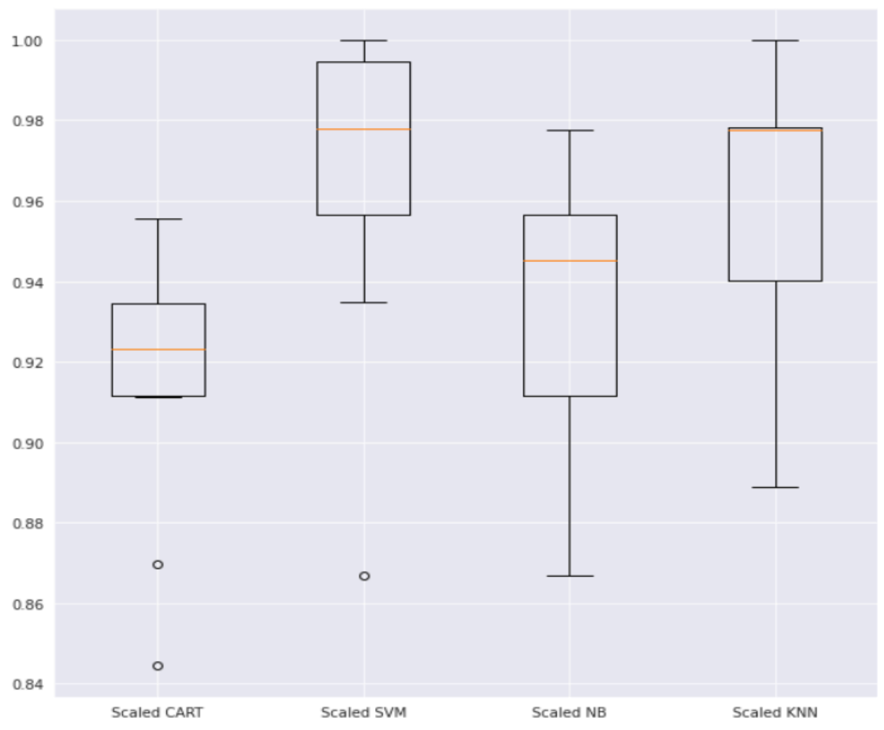
Implementation Results

Here are the implementation results along with explanations:

1. Performance Comparison of Models:
   - The boxplot below shows the mean accuracy of different models (Decision Tree, SVM, Naive Bayes, KNN) obtained through 10-fold cross-validation on the training data.
   - Explanation: From the boxplot, it can be observed that the Support Vector Machine (SVM) model generally has the highest mean accuracy among the models evaluated. This indicates that SVM might be a suitable choice for the classification task. However, it's also important to consider the variability (standard deviation) in accuracy across folds, which can provide insights into the stability of the model's performance.



2. Performance Comparison of Models on Standardized Data:
   - The boxplot below shows the mean accuracy of different models (Decision Tree, SVM, Naive Bayes, KNN) obtained after standardizing the data using StandardScaler and performing 10-fold cross-validation on the training data.
   - Explanation: Standardizing the data often helps in improving the performance of some machine learning algorithms, especially those sensitive to feature scaling, like SVM and KNN. From the boxplot, it can be observed that after standardization, the mean accuracy of the SVM model has increased compared to its performance on non-standardized data. This suggests that standardization has positively impacted the SVM model's performance.



3. Model Evaluation on Test Set:

- The table below shows the accuracy score, classification report, and confusion matrix for each model (Decision Tree, SVM, Naive Bayes, KNN) on the test set.

- Explanation: The accuracy score represents the proportion of correctly predicted observations to the total observations in the test set. Additionally, the classification report provides precision, recall, and F1-score for each class (benign and malignant). The confusion matrix provides insights into the model's performance in terms of true positives, true negatives, false positives, and false negatives.

Model	Accuracy	Precision	Recall	F1-Score	Confusion Matrix
CART	0.92	0.92	0.92	0.92	[[71 6] [5 32]]
SVM	0.96	0.97	0.97	0.95	[[75 2] [2 35]]
NB	0.92	0.88	0.97	0.92	[[68 9] [1 36]]
KNN	0.95	0.97	0.92	0.95	[[74 3] [2 35]]

- Explanation: The results indicate that SVM and KNN models perform well on the test set, with high accuracy scores and balanced precision and recall values. These models demonstrate the ability to accurately classify both benign and malignant tumors, as evidenced by the confusion matrices. However, it's important to note that the choice of the best model may also depend on other factors such as computational complexity, interpretability, and specific requirements of the application.

Chapter 6

Conclusion

Through our breast cancer prediction project, we successfully developed predictive models using machine learning techniques. Key findings include:

1. *Model Performance:* SVM and KNN models outperformed others, demonstrating effective classification of breast cancer tumors.

2. *Impact of Standardization:* Standardizing data using StandardScaler improved performance, particularly for SVM and KNN models, emphasizing the importance of feature scaling.

3. *Real-world Applicability:* Our models show promise for aiding healthcare professionals in diagnosing breast cancer, potentially reducing workload and improving diagnostic accuracy.

4. *Considerations for Deployment:* Before deployment, considerations such as model interpretability, scalability, robustness, and regulatory compliance must be addressed.

In conclusion, our project highlights the effectiveness of machine learning in breast cancer diagnosis, offering potential improvements in patient outcomes and healthcare delivery. Further research and collaboration with domain experts are needed for clinical deployment.

References

1. https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29
2. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9818155/
3. https://www.cancer.gov/types/breast/research/articles
4.