NAME: ThankGod Akornuche ThankGod

STUDENT ID: 23037727

COURSE TITLE: Machine learning

GITHUB LINK: https://github.com/Teegee2000/EVALUATING-SUPERVISED-AND-ENSEMBLE-MACHINE-LEARNING-TECHNIQUES-FOR-HEART-DISEASE-PREDICTION

EVALUATING SUPERVISED AND ENSEMBLE MACHINE LEARNING TECHNIQUES FOR HEART DISEASE PREDICTION

ABSTRACT

This study compares ensemble methods including Gradient Boosting, Bagging, Stacking, and Voting with supervised learning models like K-Nearest Neighbours (KNN), Support Vector Classifier (SVC), Logistic Regression (LR), and Gaussian Naive Bayes (GNB). The goal is to assess how well they predict heart disease, a serious worldwide health concern. Ensemble models are unique in that they can include several classifiers, reducing the shortcomings of individual models while increasing accuracy and robustness. This investigation illustrates the efficacy of machine learning approaches in healthcare diagnostics using the prediction of heart disease as a case study.

1. INTRODUCTION

Millions of deaths worldwide are attributed to heart disease each year, making it one of the leading causes of mortality. Its destructive effects on people and healthcare systems can be lessened with early identification and action. By using data to find patterns that conventional approaches frequently overlook, machine learning has become a potent tool for heart disease prediction. These models make precise and fast predictions by using clinical characteristics, diagnostic data, and patient demographics.

Supervised learning models like K-Nearest Neighbors (KNN), Support Vector Classifier (SVC), Logistic Regression (LR), and Gaussian Naive Bayes (GNB) are widely employed for classification tasks. While effective in many scenarios, these models often face challenges such as overfitting, under fitting, and inability to capture complex relationships in data.

By integrating predictions from several models, ensemble learning approaches overcome these drawbacks and increase overall accuracy and robustness. By combining various classifiers, techniques like Gradient Boosting, Bagging, Stacking, and Voting lower variance and increase predictive power. Using the prediction of heart disease as a case study, this paper assesses and contrasts these supervised and ensemble techniques. The goal is to show how ensemble approaches perform more accurately and reliably than standard models, which helps improve healthcare decision-making.

2. LITERATURE REVIEW

Machine learning approaches for heart disease prediction have been thoroughly investigated, with developments aimed at enhancing robustness and accuracy. Traditional methods like Support Vector Machines (SVM) and Logistic Regression were used in early efforts; these models had a moderate amount of success but had trouble managing noisy datasets and identifying complex relationships. (Karna et al., 2024)

The use of ensemble approaches to improve performance has been highlighted by recent studies. presented majority-voting ensembles with an accuracy of 88.5% that used SVM, Decision Trees, and Naive Bayes. In a similar vein, Tamar and Agarwal highlighted the effectiveness of Least Square Twin SVM in managing high-dimensional datasets by enhancing heart disease predictions through the use of feature selection approaches. (Swathisree et al., 2024)

Because of its capacity to lessen bias and variance, gradient boosting and xg boost have become more and more popular. Significant increases in sensitivity and specificity were observed in a study that combined these techniques with feature selection. Additionally, by combining various models, stacking and bagging have shown great accuracy in predicting cardiac disease, utilizing the advantages of several algorithms to offset their shortcomings.(Priya et al., 2023)

There are still issues, such as managing unbalanced datasets, missing values, and processing requirements. Notwithstanding these drawbacks, ensemble approaches have continuously outperformed other approaches, allowing for more accurate predictions in clinical settings. Improvements were made to increase robustness and accuracy. Early research used conventional models such as Support Vector Machines (SVM) and Logistic Regression, which had mediocre results but had trouble managing noisy datasets and capturing intricate correlations. .(Karna et al., 2024)

2.1 MACHINE LEARNING OVERVIEW

In the field of artificial intelligence known as machine learning (ML), algorithms are trained to recognize patterns in data and provide predictions. In order for models to learn the relationships between inputs (features) and outputs (labels), datasets must be fed into them. ML can be divided into three categories: supervised, unsupervised, and reinforcement learning. For structured tasks like heart disease prediction, supervised and ensemble learning are essential.

SUPERVISED MACHINE LEARNING

Supervised learning involves training a model on labeled data, where the input features (e.g., age, cholesterol level) and their corresponding output labels (e.g., heart disease presence) are provided. The goal is to learn a mapping function that can predict outcomes for unseen data.

a. ENSEMBLED MACHINE LEARNING

Ensemble learning combines multiple base models to enhance the predictive performance of a machine learning system. Unlike single models, ensembles harness diversity to improve robustness and generalizability.

WHAT IS A BASE MODEL?

An individual learning algorithm that functions as a part of an ensemble is called a base model. Every base model generates its own predictions and works independently on the dataset. For instance:

Several decision trees (base models) are trained using bootstrapped datasets in the Bagging process.

The independence and diversity of an ensemble's base models have a significant impact on its performance. Their combined predictions might not provide appreciable gains over a single model if the basic models are too similar. On the other hand, by capturing various facets of the data, varied base models can improve generalizability and lower mistakes.

3. DATASET DESCRIPTION

The dataset used contains 1025 samples with 14 features including age, sex, cholesterol levels, blood pressure, and more. The target variable indicates the presence (1) or absence (0) of heart disease.

```
df.shape
```

```
(1025, 14)
```

Figure 1: size of the dataset

3.1 DATASET AND EXPLORATORY DATA ANALYSIS (EDA)

The dataset used was downloaded from kaggle, it contains clinical and demographic attributes, such as age, cholesterol level, and resting blood pressure, and a target variable indicating heart disease presence (1) or absence (0). Key EDA steps:

Overview: Displayed dataset structure, summary statistics, and data types using .info() and .describe().

```
[1]: import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
     import numpy as np

[2]: df=pd.read_csv('heart.csv')

[3]: df
```

[3]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1020	59	1	1	140	221	0	1	164	1	0.0	2	0	2	1
1021	60	1	0	125	258	0	0	141	1	2.8	1	1	3	0
1022	47	1	0	110	275	0	0	118	1	1.0	1	1	2	0
1023	50	0	0	110	254	0	0	159	0	0.0	2	0	2	1
1024	54	1	0	120	188	0	1	113	0	1.4	1	1	3	0

1025 rows × 14 columns

Figure 2: Heart disease dataset.

1. **Missing Values**: Checked for missing data using .isnull().sum().

```
df.isnull().sum()
```

```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
```

Figure 3: checking missing data.

2. **Visualizations:**

Bar plots for categorical variables (gender).

```
## for male
male=len(df[df['sex']==0])
```

```
male
```

```
312
```

```
## for female
female=len(df[df['sex']==1])
```

```
female
```

```
713
```

```
[4]: ## visiualizing based on CLass and age

print(df.groupby(['sex'])['sex'].count())
df.groupby(['sex'])['sex'].count().plot(kind="bar")
plt.show()
```

```
sex
0    312
1    713
Name: sex, dtype: int64
```

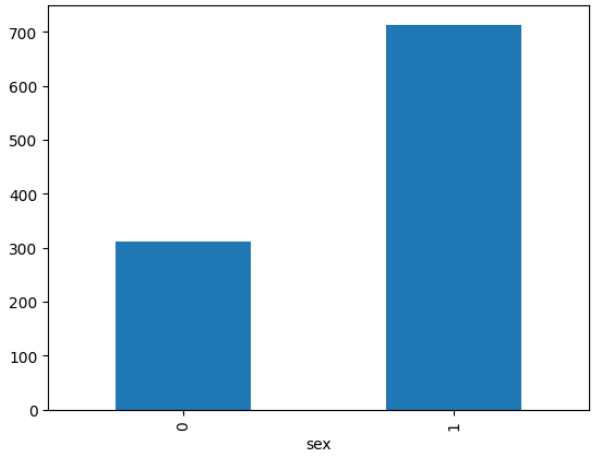


Figure 4: visualizing the gender in the dataset.

3.1 TARGET VARIABLE

A target variable, also known as the dependent variable, is the output or label the model aims to predict based on input features. In heart disease prediction, the target variable indicates the presence (1) or absence (0) of heart disease.

```
[11]: X=df.drop('target',axis=1)
y=df['target']
```

Figure 5: Target variable

3.2 DATA PREPROCESSING

Splitting Data: 80% for training and 20% for testing.

Feature Scaling: Standardized features for models sensitive to scale.

```
[12]: ## let's split the dataframe for training and testing

from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42)
```

Figure 6: splitting dataset into train and test.

3.3 CONFUSION MATRIX

A confusion matrix is a performance metric used to evaluate classification models. It provides a tabular representation of actual vs. predicted values, including:

- **True Positives (TP)**: Correctly predicted positive cases.
- **True Negatives (TN)**: Correctly predicted negative cases.
- **False Positives (FP):** Incorrectly predicted positive cases.
- **False Negatives (FN):** Incorrectly predicted negative cases.

The confusion matrix helps assess the model's accuracy, precision, recall, and F1 score, offering insight into where the model makes errors (e.g., misclassifying patients with heart disease as healthy).

```
from sklearn.metrics import roc_curve,auc
# Predicting the Test set results
from sklearn.metrics import confusion_matrix,accuracy_score

def report_performance(model):

    model_test = model.predict(X_test)

    print("\n\nConfusion Matrix:")
    print("{0}".format(metrics.confusion_matrix(y_test, model_test)))
    print("\n\nClassification Report: ")
    print(metrics.classification_report(y_test, model_test))
    #cm = metrics.confusion_matrix(y_test, model_test)
    plot_confusion_metrix(y_test, model_test)
```

Figure 7: confusion matrix code for the project.

3.4 RECEIVER OPERATING CHARACTERISTIC (ROC) CURVE

The ROC curve is a graphical representation of a classifier's performance across different thresholds. It plots:

- True Positive Rate (TPR): Sensitivity or recall.
- False Positive Rate (FPR): The proportion of negative cases incorrectly classified as positive.

The area under the ROC curve (AUC-ROC) provides a single metric to evaluate the model's ability to distinguish between classes. A higher AUC indicates better performance in predicting heart disease.

```
def roc_curves(model):
    predictions_test = model.predict(X_test)
    fpr, tpr, thresholds = roc_curve(predictions_test,y_test)
    roc_auc = auc(fpr, tpr)

    plt.figure()
    plt.plot(fpr, tpr, color='darkorange', lw=1, label='ROC curve (area = %0.2f)' % roc_auc)
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic')
    plt.legend(loc="lower right")
    plt.show()
```

Figure 8: Roc curve.

4.0 MACHINE LEARNING TECHNIQUES

SUPERVISED LEARNING

Supervised learning involves training models using labeled data, where each input feature corresponds to a known output label. The algorithm learns to map inputs to outputs and generalizes to unseen data.

- **Logistic Regression**: Predicts probabilities for binary classification problems. It is simple, interpretable, and performs well with linear relationships.
- **K-Nearest Neighbors (KNN):** Classifies data points based on the majority vote of 'k' nearest neighbors. Effective for simple datasets but sensitive to noise.
- **Support Vector Classifier (SVC):** Separates data using optimal hyperplanes, ideal for non-linear separable data but computationally intensive.
- **Gaussian Naive Bayes (GNB):** Assumes feature independence and uses Gaussian distributions. It is fast and efficient but may underperform when features are correlated.

ENSEMBLE LEARNING

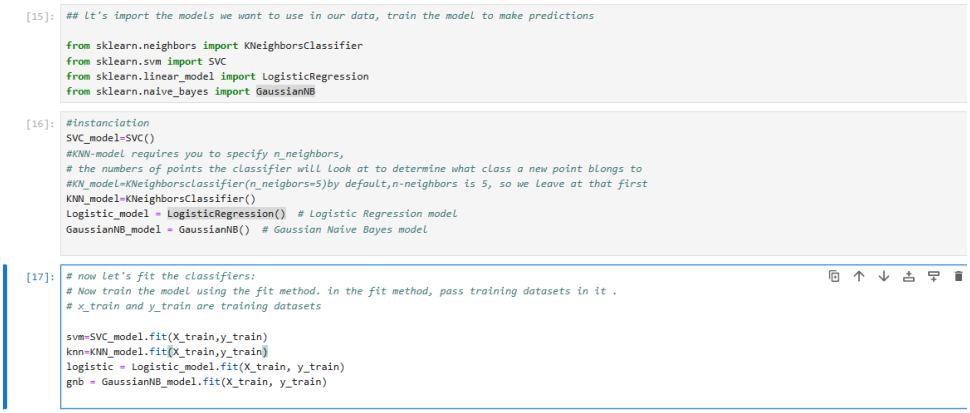
Ensemble learning improves model performance by aggregating predictions from multiple models.

- **Bagging**: Trains multiple models on bootstrapped datasets and averages predictions to reduce variance (e.g., Random Forest).
- **Boosting:** Sequentially refines weak learners to minimize errors

- **Stacking:** Combines predictions of diverse models using a meta-classifier.
- **Voting:** Aggregates predictions from multiple models using majority or weighted voting.

5.0 PERFORMANCE METRICS

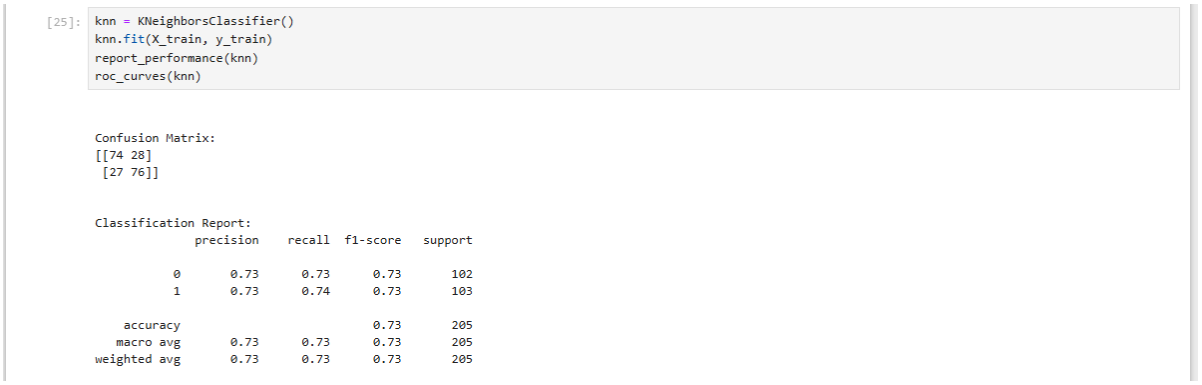
Before each model is tested it has to be trained, below is how I trained supervised machine learning model used



```
[15]: ## Lt's import the models we want to use in our data, train the model to make predictions

from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB

[16]: #instanciation
SVC_model=SVC()
#KNN-model requires you to specify n_neighbors,
# the numbers of points the classifier will look at to determine what class a new point blongs to
#KN_model=KNeighborsclassifier(n_neigbors=5)by default,n-neighbors is 5, so we leave at that first
KNN_model=KNeighborsClassifier()
Logistic_model = LogisticRegression()  # Logistic Regression model
GaussianNB_model = GaussianNB()  # Gaussian Naive Bayes model

[17]: # now let's fit the classifiers:
# Now train the model using the fit method. in the fit method, pass training datasets in it .
# x_train and y_train are training datasets

svm=SVC_model.fit(X_train,y_train)
knn=KNN_model.fit(X_train,y_train)
logistic = Logistic_model.fit(X_train, y_train)
gnb = GaussianNB_model.fit(X_train, y_train)
```

Figure 9: Training the supervised model.

1. **K-Nearest Neighbors (KNN):** A simple, non-parametric classifier that bases predictions on the closest neighbors in the feature space.



```
[25]: knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
report_performance(knn)
roc_curves(knn)


Confusion Matrix:
[[74 28]
 [27 76]]


Classification Report:
              precision    recall  f1-score   support

           0       0.73      0.73      0.73       102
           1       0.73      0.74      0.73       103

    accuracy                           0.73       205
   macro avg       0.73      0.73      0.73       205
weighted avg       0.73      0.73      0.73       205
```

Figure 10: Accuracy and Classification report for KNN model

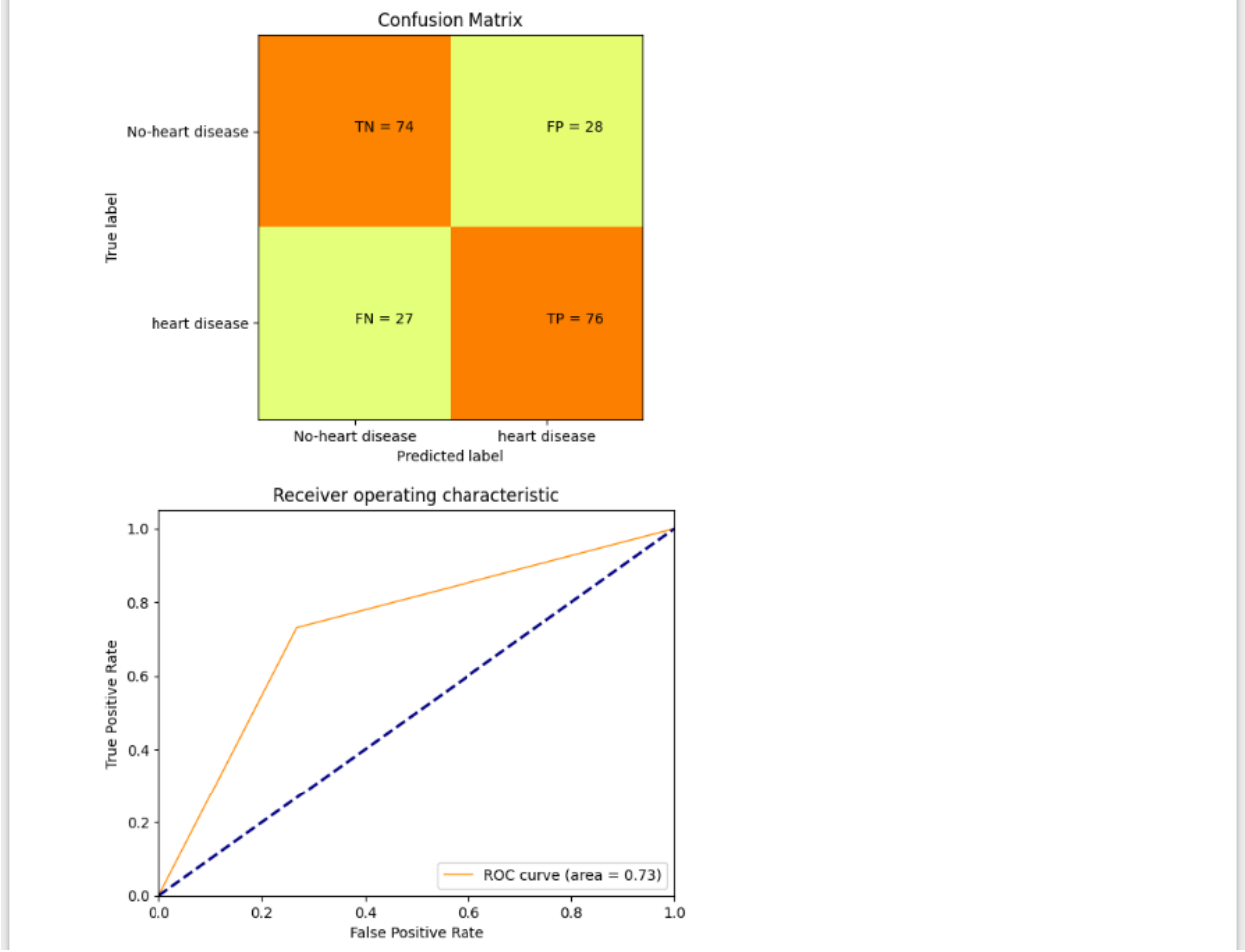


Figure 11: Confusion matrix and ROC curve for KNN model.

2. **Support Vector Classifier (SVC):** Separates classes by finding an optimal hyperplane.

```
[24]: svm = SVC()
      svm.fit(X_train, y_train)
      report_performance(svm)
      roc_curves(svm)
```

```
Confusion Matrix:
[[62 40]
 [25 78]]


Classification Report:
              precision    recall  f1-score   support

           0       0.71      0.61      0.66       102
           1       0.66      0.76      0.71       103

    accuracy                           0.68       205
   macro avg       0.69      0.68      0.68       205
weighted avg       0.69      0.68      0.68       205
```

Figure 9: classification report for SVM model.

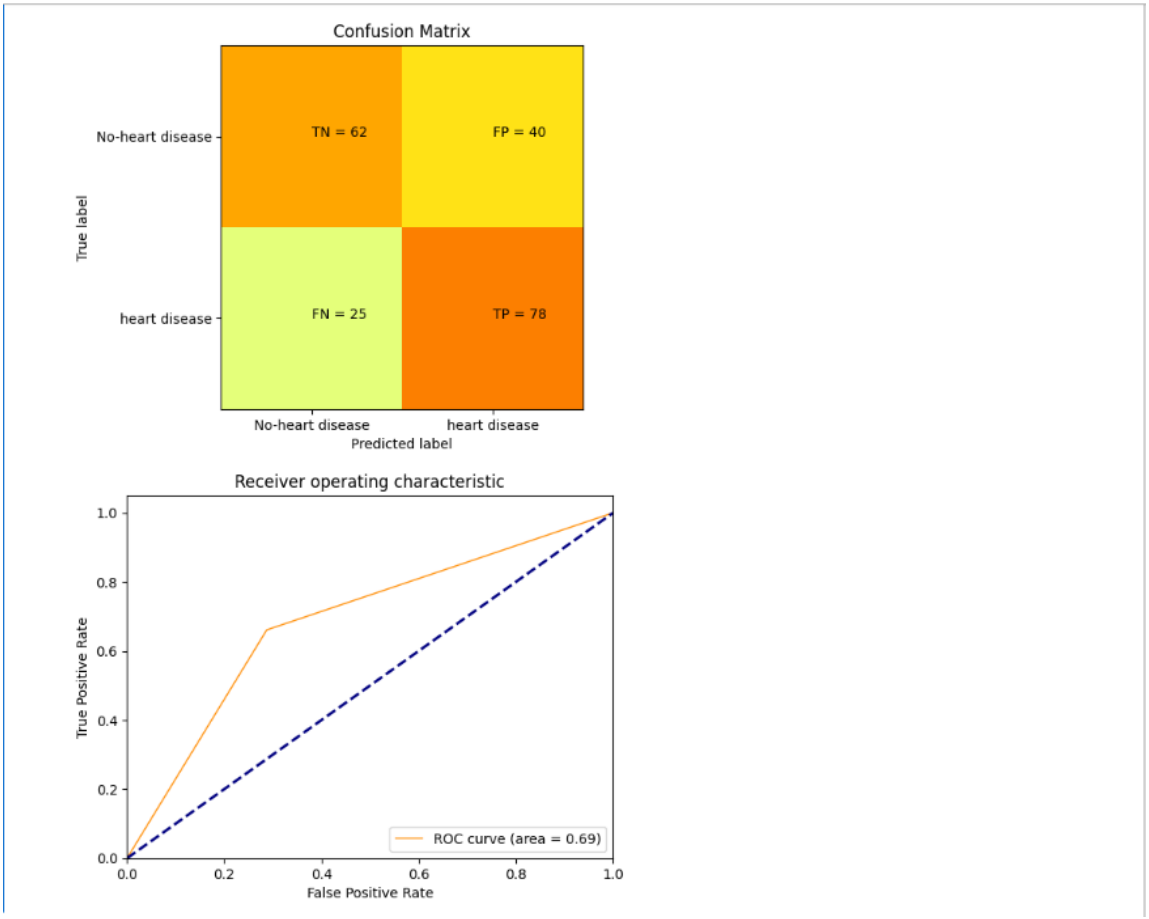


Figure 12: Confusion matrix and ROC curve for SVM model.

3. **Logistic Regression (LR):** Models the probability of a binary outcome based on input features.

```
[27]: log =LogisticRegression()
      log.fit(X_train, y_train)
      report_performance(log)
      roc_curves(log)
```

```
Confusion Matrix:
[[71 31]
 [13 90]]


Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.70      0.76       102
           1       0.74      0.87      0.80       103

    accuracy                           0.79       205
   macro avg       0.79      0.78      0.78       205
weighted avg       0.79      0.79      0.78       205
```

Figure 13: Classification report for Logistic regression

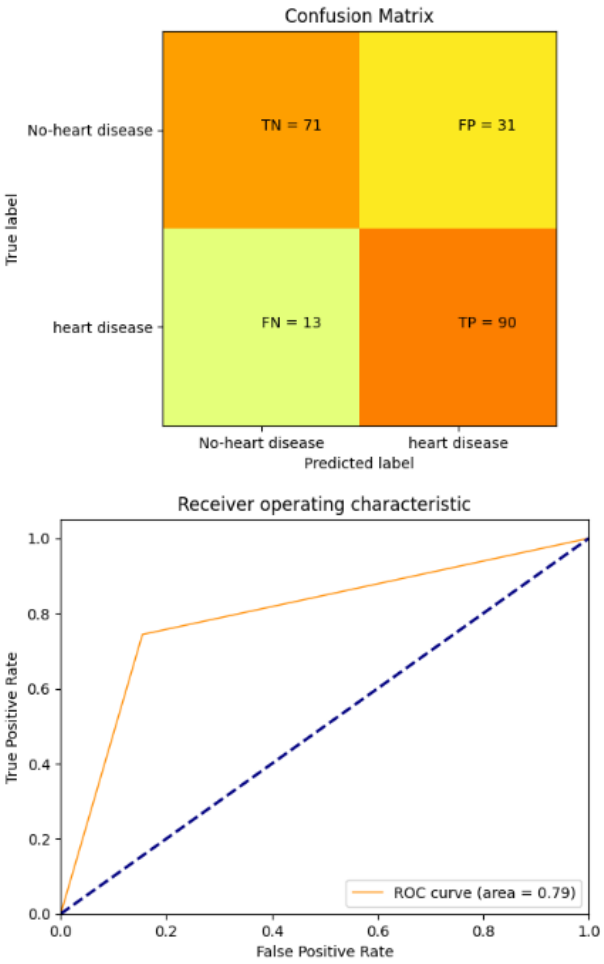


Figure 14: Confusion matrix and ROC curve for logistic regression

4. **Gaussian Naive Bayes (GNB):** Assumes Gaussian distribution for continuous features and independence between them.

```
[28]: GNB =GaussianNB()
GNB.fit(X_train, y_train)
report_performance(GNB)
roc_curves(GNB)
```

```
Confusion Matrix:
[[72 30]
 [11 92]]


Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.71      0.78       102
           1       0.75      0.89      0.82       103

    accuracy                           0.80       205
   macro avg       0.81      0.80      0.80       205
weighted avg       0.81      0.80      0.80       205
```

Figure 15: Classification report for Gaussian Naïve bayes model

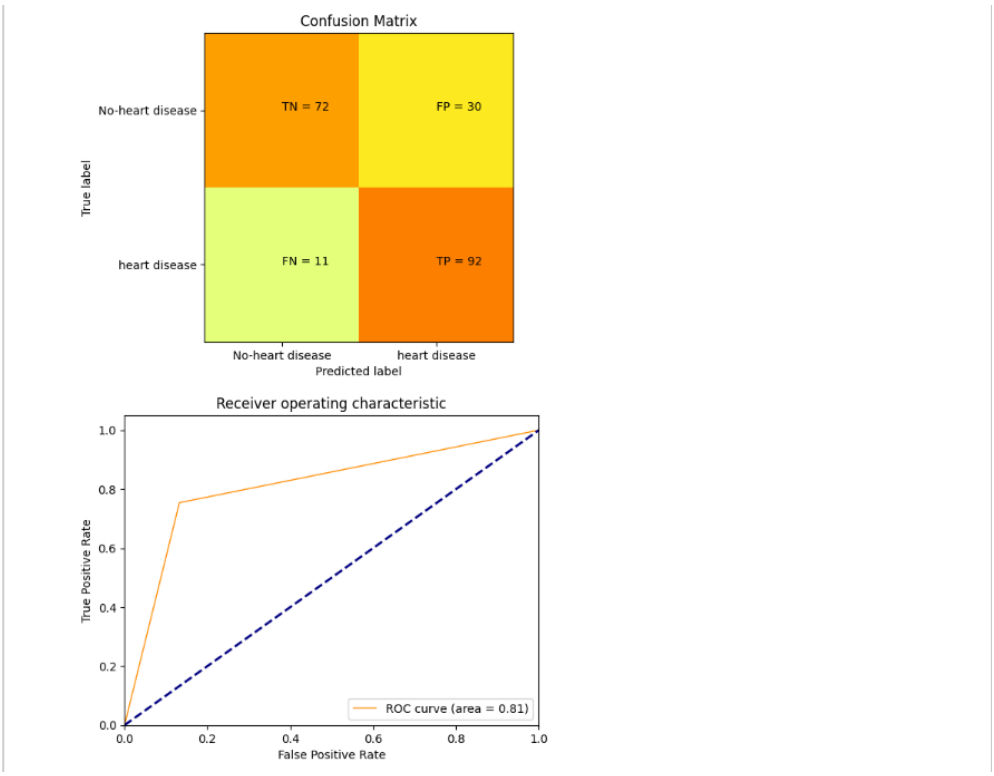


Figure 16: Confusion matrix and ROC curve for naïve bayes model

```
[31]: # Sample data for algorithm performance comparison
algorithms = ['SVM', 'KNN', 'LOG', 'GNB']
accuracy_scores = [0.68, 0.73, 0.79, 0.80]

# Plotting the comparison
plt.figure(figsize=(8, 5))
plt.bar(algorithms, accuracy_scores, color='blue')
plt.xlabel('Algorithms')
plt.ylabel('Accuracy Score')
plt.title('Comparison of Supervised Machine Learning Algorithms')
plt.ylim(0, 1)  # Set the y-axis limits between 0 and 1
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability

# Display the accuracy scores on top of the bars
for i, score in enumerate(accuracy_scores):
    plt.text(i, score, str(score), ha='center', va='bottom')

plt.tight_layout()  # Adjust the layout to prevent overlapping elements
plt.show()
```

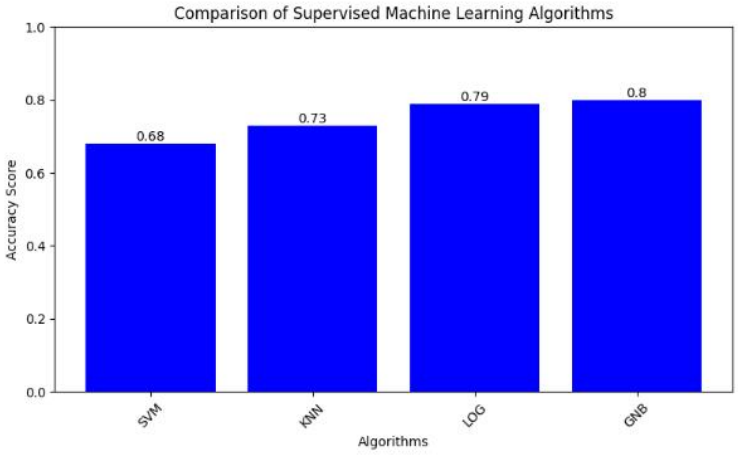


Figure 17: comparison of the supervised machine learning model used.

4.1 ENSEMBLE TECHNIQUES

Gradient Boosting: Sequentially builds weak learners to reduce errors.

```
from sklearn.ensemble import GradientBoostingClassifier
```

```
gbc=GradientBoostingClassifier()
gbc.fit(X_train,y_train)
y_pred_train_gbc=gbc.predict(X_train)
train_accuracy_gbc=accuracy_score(y_train,y_pred_train_gbc)
y_pred_gbc=gbc.predict(X_test)
accuracy=accuracy_score(y_test,y_pred_gbc)
print(' accuracy of gradient boosting',accuracy)
print('================================================')
```

```
 accuracy of gradient boosting 0.9317073170731708
================================================
```

```
gbc.fit(X_train,y_train)
pred_gbc=gbc.predict(X_test)
report_performance(gbc)
roc_curves(gbc)
```

```
Confusion Matrix:
[[93  9]
 [ 5 98]]


Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.91      0.93       102
           1       0.92      0.95      0.93       103

    accuracy                           0.93       205
   macro avg       0.93      0.93      0.93       205
weighted avg       0.93      0.93      0.93       205
```

Figure18: classification report for gradient boosting classifier

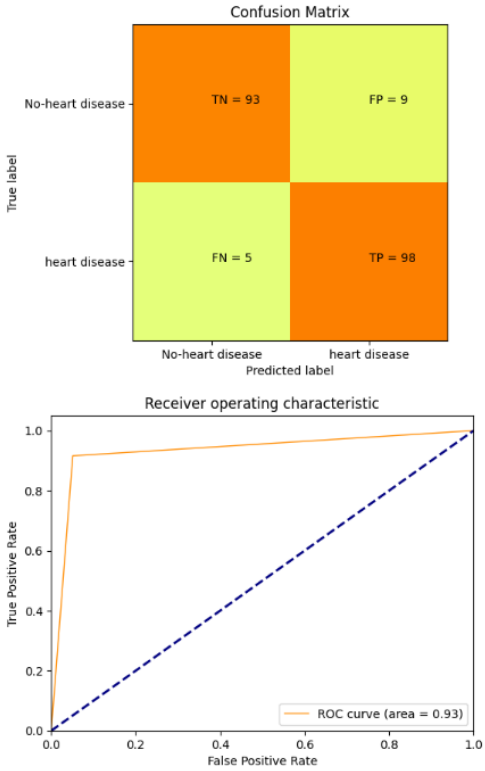


Figure 19: confusion matrix and ROC curve for gradient boosting

Bagging: Combines predictions of multiple independent models to reduce variance.

```
from sklearn.ensemble import BaggingClassifier
```

```
bc= BaggingClassifier()
bc.fit(X_train,y_train)
y_pred_train_bc=bc.predict(X_train)
train_accuracy_bc=accuracy_score(y_train,y_pred_train_bc)
y_pred_bc=bc.predict(X_test)
accuracy1=accuracy_score(y_test,y_pred_bc)
print('accuracy of bagging',accuracy1)
print('================================================')
```

```
accuracy of bagging 0.9707317073170731
================================================
```

```
bc.fit(X_train,y_train)
pred_bc=bc.predict(X_test)
report_performance(bc)
roc_curves(bc)
```

```
Confusion Matrix:
[[102   0]
 [  3 100]]


Classification Report:
              precision    recall  f1-score   support

           0       0.97      1.00      0.99       102
           1       1.00      0.97      0.99       103

    accuracy                           0.99       205
   macro avg       0.99      0.99      0.99       205
weighted avg       0.99      0.99      0.99       205
```

Figure 20: classification report for bagging classifier

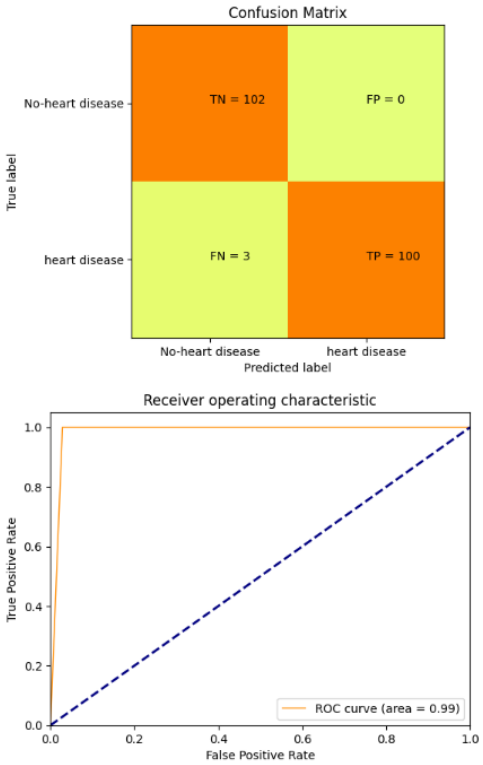


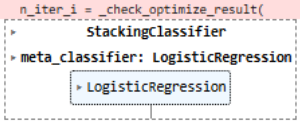
Figure 21: Confusion matrix and ROC curve for bagging classifier.

Stacking: Leverages a meta-model to aggregate predictions from base classifiers.

```
[35]: from sklearn.metrics import roc_auc_score as roc
      from sklearn.ensemble import RandomForestClassifier
      from sklearn.linear_model import LogisticRegression
      from mlxtend.classifier import StackingClassifier

      # Create individual classifiers
      clf1 = RandomForestClassifier()
      clf2 = LogisticRegression()

      # Create the stacking classifier with a list of classifiers and a meta-classifier
      stacking_clf = StackingClassifier(classifiers=[clf1, clf2], meta_classifier=clf2)

[36]: # Fit the stacking classifier on the training data
      stacking_clf.fit(X_train, y_train)
```

```
C:\Users\SBMCODED\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to
converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

36]:




```
Confusion Matrix:
[[102   0]
 [  3 100]]


Classification Report:
              precision    recall  f1-score   support

           0       0.97      1.00      0.99       102
           1       1.00      0.97      0.99       103

    accuracy                           0.99       205
   macro avg       0.99      0.99      0.99       205
weighted avg       0.99      0.99      0.99       205
```

Figure 22: Based model and classification report for stacking classifier

Based model used for stacking

1. Random forest classifier
2. Logistic regression

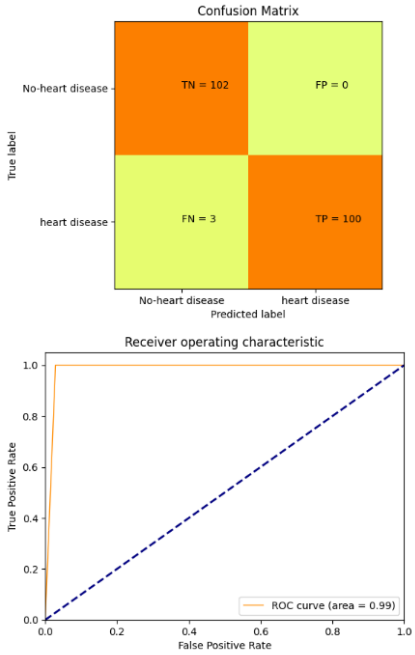


Figure 23: confusion matrix and ROC curve for stacking classifier

Voting: Combines predictions using majority or weighted voting.

```
[40]: from sklearn.ensemble import RandomForestClassifier, VotingClassifier
      from sklearn.linear_model import LogisticRegression
      from sklearn.model_selection import train_test_split
      from sklearn.metrics import accuracy_score

[41]: # Create the Voting Classifier with a list of base classifiers and the voting method
      voting_clf = VotingClassifier(estimators=[('rf', clf1), ('lr', clf2)], voting='hard')

[42]: # Fit the Voting Classifier on the training data
      voting_clf.fit(X_train, y_train)
```

```
[42]: VotingClassifier
        rf: RandomForestClassifier
        lr: LogisticRegression
```

```
[43]: # Evaluate the performance
      accuracy = accuracy_score(y_test, y_pred)
      print("Accuracy:", accuracy)

      Accuracy: 0.9853658536585366
```

```
[44]: voting_clf.fit(X_train,y_train)
      pred_vtc=voting_clf.predict(X_test)
      report_performance(voting_clf)
      roc_curves(voting_clf)
```

```
Confusion Matrix:
[[102   0]
 [ 16  87]]


Classification Report:
              precision    recall  f1-score   support

           0       0.86      1.00      0.93       102
           1       1.00      0.84      0.92       103

    accuracy                           0.92       205
   macro avg       0.93      0.92      0.92       205
weighted avg       0.93      0.92      0.92       205
```

Figure 24: Based model and classification report for voting classifier

Based model includes:

1. Random forest
2. Logistic regression

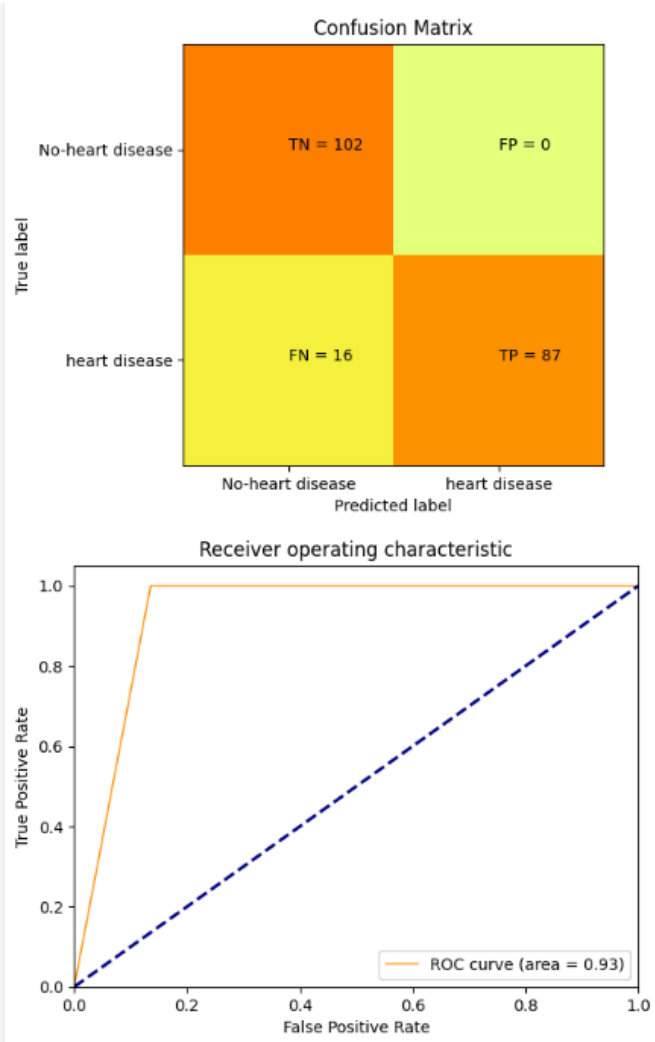


Figure 25: confusion matrix and ROC curve for voting classifier

```
[29]: # Sample data for algorithm performance comparison
algorithms = ['gradientboosting_classifier', 'Bagging_classifier', 'stacking_classifier', 'voting_classifier']
accuracy_scores = [0.93, 0.99, 0.99, 0.92]

# Plotting the comparison
plt.figure(figsize=(8, 5))
plt.bar(algorithms, accuracy_scores, color='green')
plt.xlabel('Algorithms')
plt.ylabel('Accuracy Score')
plt.title('Comparison ofensemble Machine Learning Algorithms')
plt.ylim(0, 1)  # Set the y-axis limits between 0 and 1
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability

# Display the accuracy scores on top of the bars
for i, score in enumerate(accuracy_scores):
    plt.text(i, score, str(score), ha='center', va='bottom')

plt.tight_layout()  # Adjust the layout to prevent overlapping elements
plt.show()
```

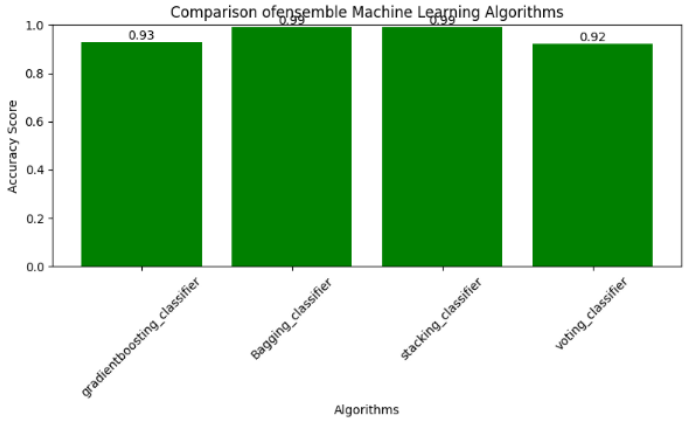


Figure 26: comparison of the ensemble machine learning model.

5. COMPARATIVE ANALYSIS

For simple datasets, supervised learning models such as Gaussian Naive Bayes, KNN, SVC, and logistic regression are appropriate since they are easy to use and understand. But they frequently have trouble with intricate interactions, noise, and overfitting.

By merging predictions from several models, ensemble learning approaches get beyond these restrictions. While Boosting minimizes bias through iterative learning, Bagging decreases variance. Voting produces forecasts that are balanced, while stacking uses a variety of models to increase accuracy. Because ensemble approaches combine the advantages of many techniques, they routinely perform better than supervised models and provide increased accuracy and robustness for the prediction of heart disease.

```
[33]: # Sample data for algorithm performance comparison
algorithms = ['SVM', 'KNN', 'LOG', 'GNB','gradientboosting_classifier', 'Bagging_classifier', 'stacking_classifier', 'voting_classifier']
accuracy_scores = [0.68, 0.73, 0.79, 0.80,0.93, 0.99, 0.99, 0.92]

# Plotting the comparison
plt.figure(figsize=(8, 5))
plt.bar(algorithms, accuracy_scores, color='pink')
plt.xlabel('Algorithms')
plt.ylabel('Accuracy Score')
plt.title('evaluating supervised and ensemble Machine Learning Algorithms')
plt.ylim(0, 1)  # Set the y-axis limits between 0 and 1
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability

# Display the accuracy scores on top of the bars
for i, score in enumerate(accuracy_scores):
    plt.text(i, score, str(score), ha='center', va='bottom')

plt.tight_layout()  # Adjust the layout to prevent overlapping elements
plt.show()
```

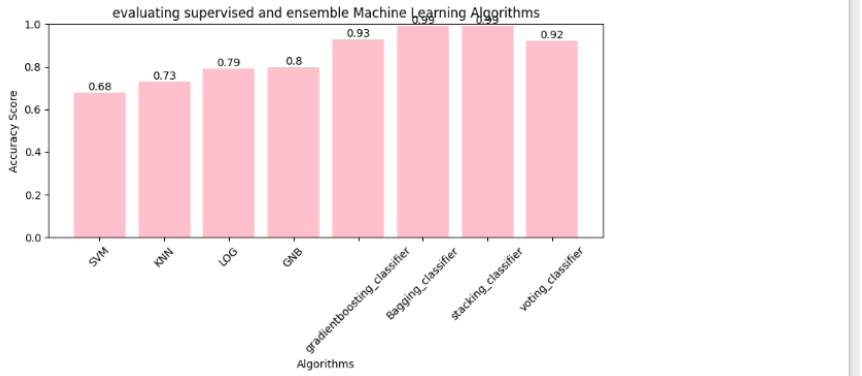


Figure 27: comparative analysis between machine learning and ensemble model.

6. RESULTS AND DISCUSSION

- **Supervised Models**: Logistic Regression achieved 79% accuracy, while KNN and SVC showed moderate performance.
- **Ensemble Models**: Gradient Boosting and Bagging outperformed individual models with accuracies of 93% and 99%, respectively. Stacking provided the best performance by leveraging diverse base models.

6.1 STRENGTHS OF THE APPROACH

1.**High Accuracy:** Ensemble models like Bagging and Stacking provide higher accuracy than standalone supervised models.

2. **Robustness:** Less prone to overfitting and better generalization to unseen data.

3. **Flexibility**: Ensemble methods can incorporate diverse base models, enhancing predictive capabilities.

4. **Scalability:** Effective for complex, high-dimensional datasets commonly found in healthcare.

6.2 CHALLENGES

- **Computational Complexity**: Ensemble models, particularly Stacking and Boosting, require significant computational resources.
- **Implementation Complexity**: Combining models and tuning hyperparameters in ensembles is more challenging than using standalone models.
- **Interpretability**: The aggregated predictions of ensembles can make results harder to explain compared to simpler supervised models.

6.3 FUTURE DIRECTIONS

1**. Dataset Diversity**: Expanding datasets to include more diverse patient populations will enhance model generalization.

2. **Hybrid Models**: Combining ensemble techniques with neural networks for better performance on complex tasks.

3. **Automated Machine Learning (AutoML):** Employing AutoML frameworks to optimize hyperparameters and select the best models.

7. CONCLUSION

The strengths and limitations of supervised and ensemble machine learning approaches for heart disease prediction were examined in this study. By utilizing the complementing advantages of several algorithms, ensemble models such as Gradient Boosting, Bagging, and Stacking consistently outperformed solo supervised models. This demonstrates how ensemble approaches may be used to handle complicated and noisy datasets that are frequently seen in the healthcare industry. The comparative research highlights how crucial it is to use reliable methods to improve accuracy and dependability, which makes ensemble learning a crucial instrument for clinical diagnosis and decision-making.

References

Karna, V. V. R., Karna, V. R., Janamala, V., Devana, V. N. K. R., Ch, V. R. S., & Tummala, A. B. (2024). A Comprehensive Review on Heart Disease Risk Prediction using Machine Learning and Deep Learning Algorithms. In Archives of Computational Methods in Engineering (Issue October). Springer Netherlands. https://doi.org/10.1007/s11831-024-10194-4

Priya, M., Anitha, A., Nallakaruppan, M. K., Raju, J., Raghavan, R., & Srivastava, D. (2023). Heart Disease Prediction Using Machine Learning Algorithms. 2023 Innovations in Power and Advanced Computing Technologies, i-PACT 2023, 01122. https://doi.org/10.1109/I-PACT58649.2023.10434931

Swathisree, M., Gayathri, E., & Manivannan, R. (2024). Heart Disease Prediction Using Machine Learning. Proceedings of 9th International Conference on Science, Technology, Engineering and Mathematics: The Role of Emerging Technologies in Digital Transformation, ICONSTEM 2024, March. https://doi.org/10.1109/ICONSTEM60960.2024.10568792

Links

1. https://www.researchgate.net/publication/382530262_Heart_Disease_Prediction_Using_Machine_Learning?_tp=eyJjb250ZXh0Ijp7ImZpcnN0UGFnZSI6ImhvbWUiLCJwYWdlIjoic2VhcmNoIiwicG9zaXRpb24iOiJwYWdlSGVhZGVyIn19
2. https://www.researchgate.net/publication/385012222_A_Comprehensive_Review_on_Heart_Disease_Risk_Prediction_using_Machine_Learning_and_Deep_Learning_Algorithms
3. https://www.researchgate.net/publication/379050027_Heart_disease_prediction_using_machine_learning_algorithms?_tp=eyJjb250ZXh0Ijp7ImZpcnN0UGFnZSI6ImhvbWUiLCJwYWdlIjoic2VhcmNoIiwicG9zaXRpb24iOiJwYWdlSGVhZGVyIn19