HEART DISEASE DIAGNOSIS USING RANDOM FOREST

Option 2

Q: Can a Random Forest model achieve >90% recall for heart disease screening using only primary care data?

Goal: develop a screening tool that identifies at-risk patients early (recall is more important than overall accuracy)

Training Input: 303 patient records, 13 clinical features (eg: age, cholesterol levels, chest pain type, etc.), and the target variable (num)

Training Output: Optimized Random Forest model with tuned hyperparameters

Deployment Input: new patient records with the 13 clinical features

Deployment Output: target variable (num) in binary (1=disease (aggregated from 1, 2, 3, and 4), 0=no disease)

Data Preprocessing

- Binary Convertion (for the target attribute)
- Data Splitting (train, test, validation (70, 20, 10) and stratified sampling)
- Missing Value (Mode)
- Label Encoding

Model Training:

```python
# Create a Random Forest classifier with medically-appropriate parameters
rf_model = RandomForestClassifier(
    n_estimators=100,        # Number of decision trees in the forest
    max_depth=10,            # Maximum depth of each tree
    min_samples_split=5,     # Minimum samples required to split a node
    min_samples_leaf=2,      # Minimum samples required at a leaf node
    max_features='sqrt',     # Number of features to consider for splitting
    bootstrap=True,          # Use bootstrap sampling (bagging)
    random_state=42,         # For reproducible results
    class_weight='balanced', # Adjust for class imbalance
    verbose=1                # Show training progress
)

#Train the model
# Fitting the model to learn patterns from 212 training patients
# Each tree will learn different aspects of heart disease predictors
rf_model.fit(X_train_clean, y_train)
```

```
[Parallel(n_jobs=1)]: Done  49 tasks      | elapsed:    0.1s
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    0.2s finished
```

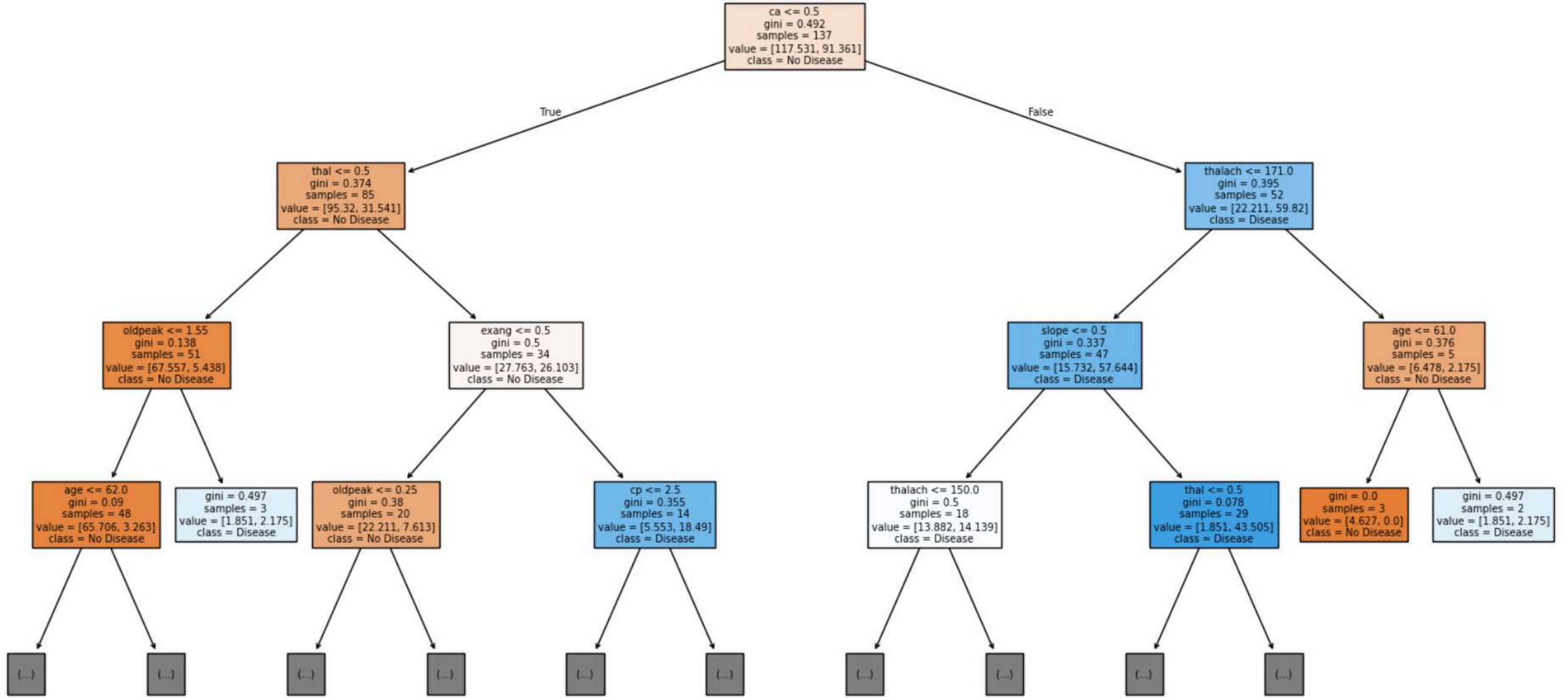
RandomForestClassifier

```
RandomForestClassifier(class_weight='balanced', max_depth=10,
                       min_samples_leaf=2, min_samples_split=5, random_state=42,
                       verbose=1)
```

164 healthy, 139 diseased (though not too much of difference, but just to be safe, balanced is used because this is medical case)

Loss function: Gini Impurity **(not specified, but by default RF use gini)**

Example Decision Tree within the Random Forest



Challenge: Overfitting

Training Set Performance	Validation Set Performance	Performance Gap
.Accuracy: 97.16%	Accuracy: 83.61%	Accuracy: 13.55%
Precision: 97.89%	Precision: 87.50%	Precision: 10.39%
Recall: 95.88%	Recall: 75.00%	Recall: 20.88%
F1-Score: 96.88%	F1-Score: 80.77%	F1-Score: 16.11%
ROC-AUC: 99.67%	ROC-AUC: 91.02%	AUC-ROC: 8.65%

validation accuracy lagged behind, indicating that the ensemble had memorized training patterns rather than learning generalizable relationships

Hyperparameter Tuning using Grid Search CV

```python
# Define parameter grid focused on reduc
param_grid = {
    'n_estimators': [50, 100, 150],
    'max_depth': [5, 8, 10, None],
    'min_samples_split': [5, 10, 15],
    'min_samples_leaf': [2, 4, 6],
    'max_features': ['sqrt', 'log2']
}
```

Total Combinations: 3 x 4 x 3 x 3 x 2 = 216
5 fold CV: 216 x 5 = 1080 model trainings

Scoring prioritise **accuracy** instead of **recall**. Usually in medical case, **recall** is prioritized to minimise false negative

```python
# Perform grid search with cross-validation
grid_search = GridSearchCV(
    estimator=RandomForestClassifier(
        random_state=42,
        class_weight='balanced',  # Keep balanced for medical data
        bootstrap=True
    ),
    param_grid=param_grid,
    cv=StratifiedKFold(n_splits=5, shuffle=True, random_state=42),  # 5-fold CV
    scoring='accuracy',  # Primary metric
    n_jobs=-1,           # Use all available CPU cores to speed up computation
    verbose=1,           # Show progress
    return_train_score=True  # Get training scores to check overfitting
)

```

Best parameters found from 5 fold CV

```
BEST PARAMETERS FOUND:
  max_depth: 5
  max_features: sqrt
  min_samples_leaf: 2
  min_samples_split: 5
  n_estimators: 100
Best cross-validation accuracy: 0.8392
```

Tuning Result Analysis:

Best CV Training Accuracy Score: 0.9455 (94.55%)

Best CV Validation Accuracy Score: 0.8392 (83.92%)

Overfitting gap after tuning: 0.9455 - 0.8392 = 0.1062 (10.62%)

Overfitting gap reduced by 0.0292 (2.92%)

Aspect	Initial Model	Optimized Model	Improvement
Overfitting Gap	13.55%	10.63%	-2.92%
Training Accuracy	97.16%	94.55%	-2.61% (intentional)
CV Validation Accuracy	83.61%	83.92%	+0.31%

Top 5 parameter combinations:

```
1. Score: 0.8392 (±0.0633)
   max_depth: 5
   min_samples_split: 5
   min_samples_leaf: 2
   n_estimators: 100
   Train/Val gap: 0.1063

2. Score: 0.8392 (±0.0444)
   max_depth: 5
   min_samples_split: 15
   min_samples_leaf: 6
   n_estimators: 50
   Train/Val gap: 0.0589

3. Score: 0.8392 (±0.0633)
   max_depth: 5
   min_samples_split: 5
   min_samples_leaf: 2
   n_estimators: 100
   Train/Val gap: 0.1063

4. Score: 0.8392 (±0.0444)
   max_depth: 5
   min_samples_split: 15
   min_samples_leaf: 6
   n_estimators: 50
   Train/Val gap: 0.0589

5. Score: 0.8392 (±0.0444)
   max_depth: 8
   min_samples_split: 15
   min_samples_leaf: 2
   n_estimators: 50
   Train/Val gap: 0.0708
```

- Score accuracy the same across all 5 different parameters, not too sensitive to small parameter changes, it's robust
- Standard deviation is around 0.04 to 0.06, is stable for small dataset
- Train/val gap: 0.05 - 0.10 generelise well

```
COMPLEXITY ANALYSIS:
Average validation score for simple models (depth ≤ 8): 0.8273
Average validation score for complex models (depth > 8): 0.8258
Simpler models generalize better → Confirms overfitting issue
```

Simpler model (shallower tree) performs slightly better than complex (deeper tree). Complex model increase risk of overfitting

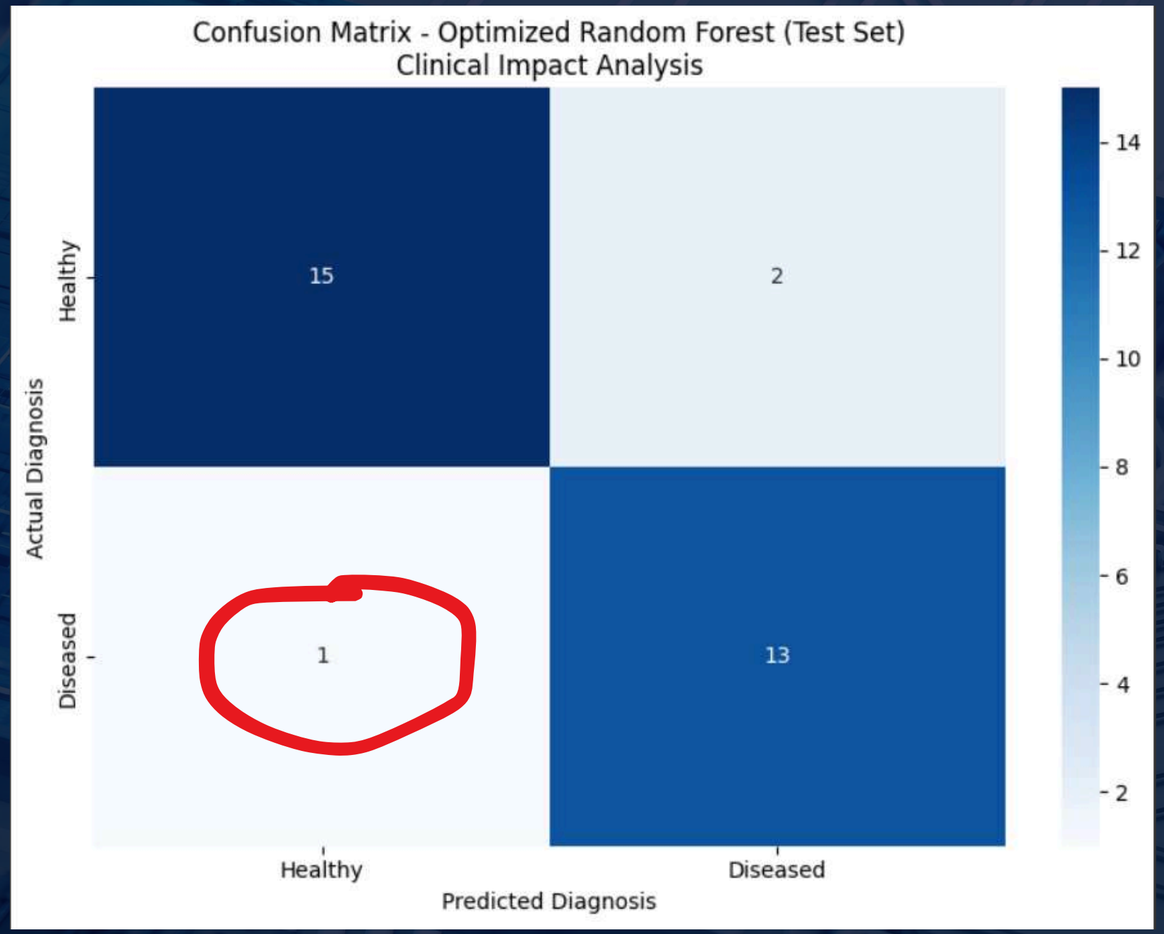
Final Results

Metric	Training Set	Validation Set	Test Set
Accuracy	92.42%	83.61%	90.32%
Precision	94.51%	87.50%	86.67%
Recall	88.66%	75.00%	92.86%
F1-Score	91.49%	80.77%	89.66%
AUC-ROC	98.83%	90.69%	94.96%

Comparison with initial model

Model	Training Acc	Validation Acc	Test Acc	Overfit Gap
Initial	97.16%	83.61%	-	13.55%
Optimized	92.42%	83.61%	90.32%	8.81%

Confusion Matrix



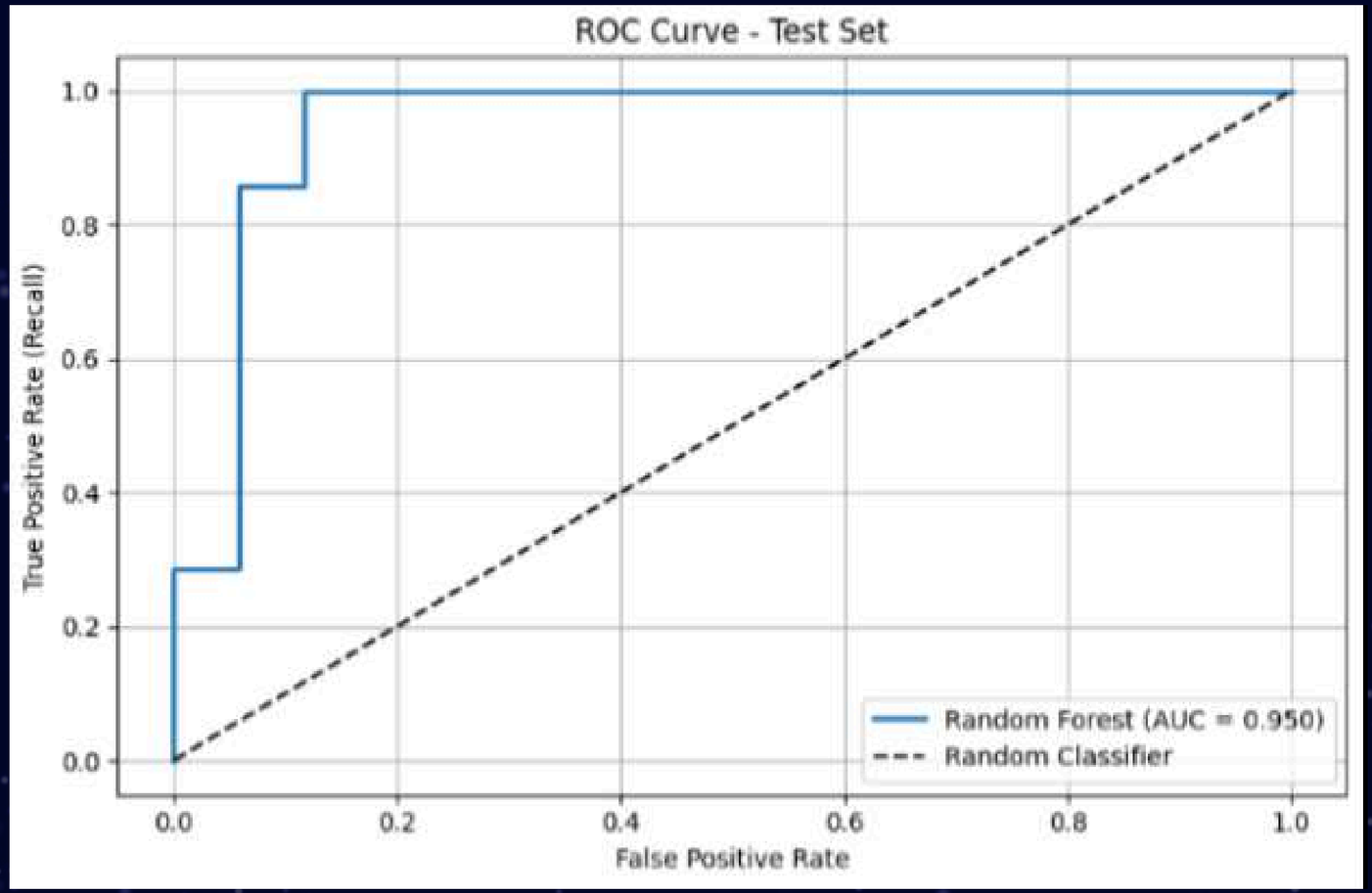
Patients correctly identified as healthy: 88.2% (15/17)

Patients correctly identified as diseased: 92.9% (13/14)

Missed disease cases (false negatives): 7.1% (1/14)

False alarms (false positives): 11.8% (2/17)

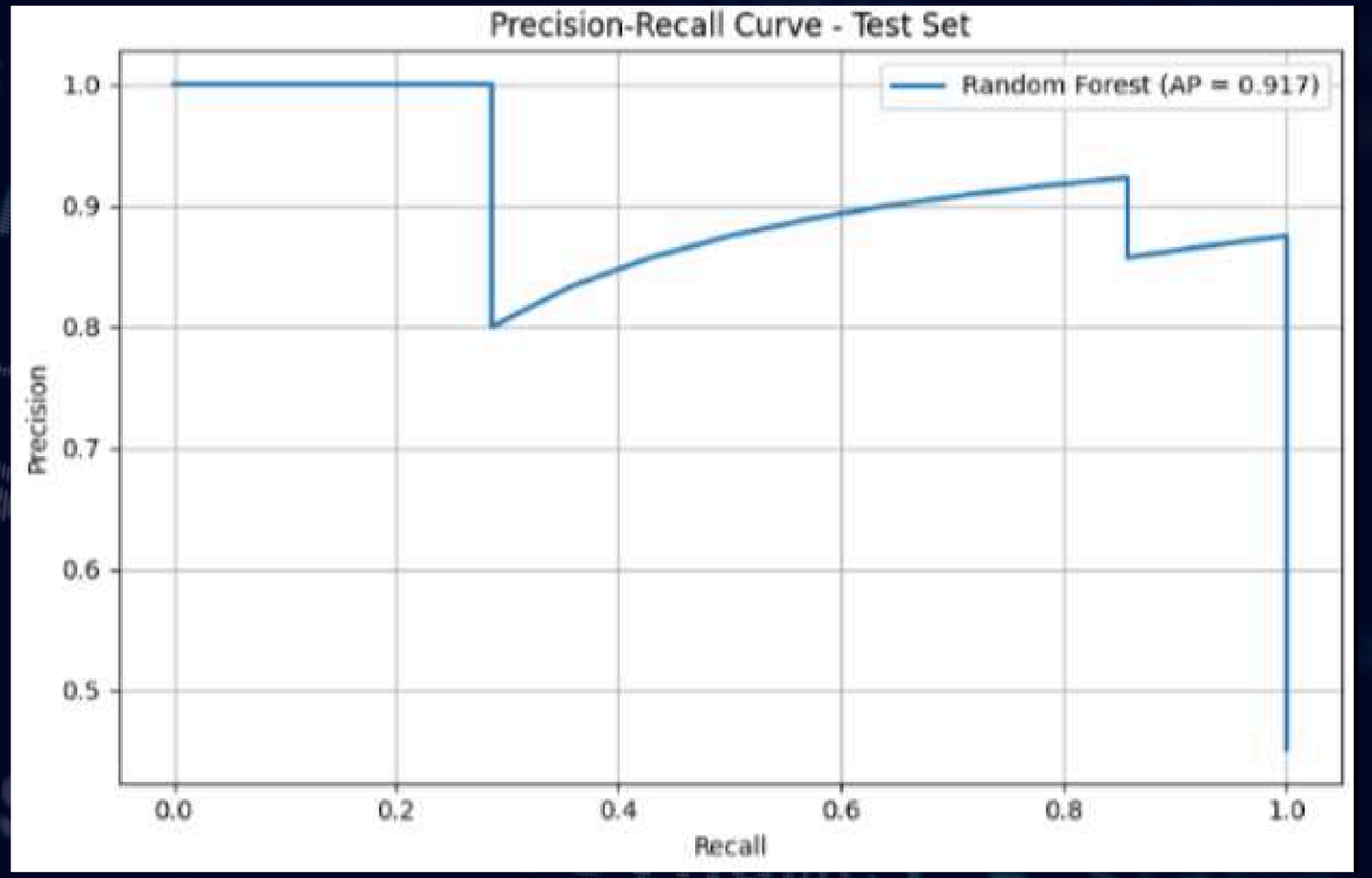
ROC-AUC



AUC = 0.950

- Model shows 95% ability to flag patients correctly by disease risk
- Surpassess the 0.90 threshold required for medical decision support tools
- Can be tuned to minimise missed cases or reduce false alarm

Precision Recall



AP (Average Precision) = 0.917

- Handles Class Imbalance: Maintains high performance despite 45.9% disease commonness
- Precision-Recall Balance: 86.67% precision at 92.86% recall
- Consistent Performance

Impact: When model predicts heart disease, can be **86.67% confident** it is correct, while **missing** only **7.1%** of actual cases

Future Direction / Works

Short-term:

- Validation data across different hospitals and populations
- Helps clinicians understand why the model make specific predictions

Medium-term:

- Enhance model to predict the severity (eg: mild, severe, mid)
- Integrate with electronic health records

Long-term:

- Accessible as a tool to help underserved communities
- Improve early detection and reduce different level of treatment among the populations

THANK YOU